Theoretical Computer Science 410 (2009) 595–613

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Computing on a partially eponymous ring☆

Marios Mavronicolas [a,b], Loizos Michael [c,*], Paul Spirakis [d,e]

[a] Department of Computer Science, University of Cyprus, Nicosia CY-1678, Cyprus
[b] Faculty of Computer Science, Electrical Engineering and Mathematics, University of Paderborn, 33102 Paderborn, Germany
[c] School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA
[d] Department of Computer Engineering and Informatics, University of Patras, 265 00 Patras, Greece
[e] Research Academic Computer Technology Institute, 265 00 Patras, Greece

ARTICLE INFO

Keywords:
Distributed computation
Partially eponymous
Ring
Solvability
Computability
Message complexity
Multiple leader election
Circularly symmetric relation

ABSTRACT

We study the *partially eponymous model* of distributed computation, which simultaneously generalizes the *anonymous* and the *eponymous* models. In this model, processors have *identities,* which are neither necessarily all identical (as in the anonymous model) nor necessarily unique (as in the eponymous model). In a *decision problem* formalized as a *relation*, processors receive *inputs* and seek to reach *outputs* respecting the relation. We focus on the *partially eponymous ring*, and we shall consider the computation of *circularly symmetric* relations on it. We consider *sets of rings* where all rings in the set have the same multiset of identity multiplicities.

- We distinguish between solvability and computability: in *solvability,* processors are required to *always* reach outputs respecting the relation; in *computability,* they must do so whenever this is possible, and must otherwise report impossibility.
  - We present a topological characterization of solvability for a relation on a set of rings, which can be expressed as an efficiently checkable, number-theoretic predicate.
  - We present a *universal* distributed algorithm for computing a relation on a set of rings; it runs *any* distributed algorithm for constructing *views,* followed by *local* steps.
- We derive, as our main result, a *universal* upper bound on the *message complexity* to compute a relation on a set of rings; this bound demonstrates a graceful degradation with the *Least Minimum Base,* a parameter indicating the degree of least possible eponymity for a set of rings. Thereafter, we identify two cases where a relation can be computed on a set of rings, with rings of size n, with an efficient number of $\mathcal{O}(n \cdot \lg n)$ messages.

1. Introduction

1.1. Motivation and framework

Two of the best studied models in Distributed Computing Theory are the *eponymous* and the *anonymous* models. Considered in both models are *decision problems* where *processors* may receive *inputs* and seek to reach *outputs* that are

☆ A preliminary version of this work appeared in the Proceedings of the 10th International Conference on Principles of Distributed Systems, A.A. Shvartsman (Ed.), pp. 380–394, in: Lecture Notes in Computer Science, vol. 4305, Springer-Verlag, December 2006. This work has been partially supported by the IST Program of the European Union under contract number 015964 (AEOLUS).

* Corresponding author.
E-mail addresses: mavronic@cs.ucy.ac.cy (M. Mavronicolas), loizos@eecs.harvard.edu (L. Michael), spirakis@cti.gr (P. Spirakis).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.10.010

admissible: they are related to the inputs according to some (*recursive*) *relation*. In both models, processors have available *identities*.

- In the eponymous model, the identities are *unique*. Such availability enables the solvability of all relations: processors first solve *Leader Election* [11] to elect a leader among them; then, the leader undertakes computation and communicates the solution to the others.
- In the anonymous model, all identities are *identical* and all processors run the same local algorithm. The impossibility of breaking this initial symmetry retains many relations unsolvable in the anonymous model; the prime example is the impossibility of solving *Leader Election* on an anonymous *ring* [1].

This long-known separation between the eponymous and the anonymous models invites the investigation of an intermediate model where the available identities are neither necessarily unique nor necessarily all identical; call it the *partially eponymous* model. Studying such a model is well motivated since it may be practically difficult to keep all identities distinct in a network with an increasing number of processors. In this work, we consider a particular case of the partially eponymous model, that of the (*asynchronous*) *partially eponymous ring* with *bidirectional* communication and *orientation*.[1]

We focus on *circularly symmetric relations*; these form the broadest class of relations that are natural to consider for rings. Roughly speaking, in a circularly symmetric relation, shifting any *output vector* that is admissible for a given *input vector* must yield an output vector that is admissible for the correspondingly shifted input vector. Circularly symmetric relations were originally motivated by the known fact that a *function* is solvable on an (asynchronous) anonymous ring if and only if the function value does not change when the inputs are cyclically shifted [2, Theorem 3.4 (Condition (i))].

A significant ingredient of some previous work on anonymous networks has been the requirement that there be a single distributed algorithm for a particular relation that runs on *all* networks and allows processors to occasionally report *impossibility* — exactly, of course, when it is impossible to return admissible outputs. This computational requirement will be called *computability* in this work; we explicitly provide the *first* formal definition of computability for a relation (Definitions 3.2 and 3.3).

An orthogonal viewpoint is to identify the subclass of networks on which it is always possible for the processors to return admissible outputs. This viewpoint follows the motivation to obtain tailored distributed algorithms that are possibly more *efficient* in terms of *message complexity* than those running on *all* networks. This requirement will be here called *solvability* (Definition 3.1); a relation is *solvable* on a set of networks if there is a distributed algorithm that runs on any network in the set so that processors always reach admissible outputs (and *never* report impossibility).

We emphasize that we do *not* view solvability as a computational requirement. Hence, we do not consider algorithms as procedures for *solving* relations; rather, we only use the notion of solvability to attest that a relation is well suited for a set of networks. Algorithms are thought of as *computing* relations on networks even if the relations *are* solvable on those networks (and thus impossibility is never reported); solvability may, of course, make computing a relation easier.

In this work, we are interested in the solvability and computability of circularly symmetric relations on partially eponymous rings. In particular, we seek a characterization of circularly symmetric relations that are solvable on particular classes of partially eponymous rings. We are also interested in the message complexity for the computability of those relations on particular classes of partially eponymous rings as a function of the ring size. (Bit complexity remains beyond the scope of this work.[2] So also do *randomized* distributed algorithms.) We mostly consider *non-uniform* distributed algorithms, where the ring size is available to the processors. Non-uniformity is known to be *essential* for computing certain functions (e.g., *Sum*) on anonymous rings [2, Theorem 3.3].

1.2. State-of-the-art

Computation on anonymous networks was first studied in the seminal work of Angluin [1], where the fundamental impossibility of solving Leader Election was first established. Yamashita and Kameda [12,13] later considered the solvability of several representative relations (such as Leader Election, *Edge Election*, *Spanning Tree Construction*, and *Topology Recognition*) on anonymous networks. For those relations, Yamashita and Kameda characterized the class of (anonymous) networks on which each relation is solvable under different assumptions on the network attributes (e.g., size, topology, etc.) that are known by the processors.

Yamashita and Kameda [14] considered the computation of functions on anonymous networks. They characterized the class of solvable functions and proved lower and upper bounds on their message complexity. Attiya et al. [2] had initiated the study of the solvability of functions on asynchronous anonymous rings.

The general model of an arbitrary, partially eponymous network was first considered by Yamashita and Kameda [15]. They focused on Leader Election and provided a graph-theoretic characterization of its solvability under different assumptions on the communication mode and the available (a)synchrony.

[1] Bidirectional communication allows messages to be sent along both directions of an edge. A ring is *oriented* if the two edges incident to each processor are consistently labelled as *left* and *right*.

[2] We note, however, that since both identities and inputs come from arbitrary domains, the bit complexity for transmitting either an identity or an input may not be bounded in terms of the ring size.

Boldi and Vigna [3] provided a general study of the solvability of an arbitrary relation on an arbitrary network, under any level of knowledge and *anonymity* (or *eponymity*) of the processors. (The detail of the level of processor anonymity is modelled in [3] by using colors on the nodes or on the edges.) Boldi and Vigna provided an *effective* (i.e., *recursive*) characterization of solvable relations on an arbitrary network; the characterization is graph-theoretic and involves the concepts of *graph coverings* and *graph fibrations* [1,4,10]. The work of Boldi and Vigna [3] has left open the possibility of devising *efficient* characterizations of solvability on particular anonymous networks, or even on (particular) partially eponymous networks.

Flocchini et al. [8, Section 2.2] considered the *Vertex Election* (identical to Leader Election), Edge Election, *General Election* (generalizing simultaneously the Vertex Election and Edge Election relations) and *Multiset Sorting* relations on asynchronous anonymous rings where processors are distinguished by *input values* that are not necessarily distinct. So, input values are treated in the partially eponymous model of Flocchini et al. [8] either as identities (e.g., when studying Vertex Election) or as inputs (e.g., when studying Multiset Sorting). We emphasize that the partially eponymous model of Flocchini et al. [8] does *not* simultaneously consider identities and inputs, while our model does.

Under the assumptions that input values are binary and the size n of the ring is prime, Flocchini et al. [8, Theorems 4.1 and 4.2] provided lower and upper bounds on message complexity for those four relations; the lower and upper bounds were $\Omega(\sum_k (z_k^2 + t_k^2))$ and $\mathcal{O}\left(\sum_k (z_k^2 + t_k^2) + n \cdot \lg n\right)$, respectively, where z_k and t_k are the lengths of consecutive blocks of 1's and 0's, respectively, in the input vector. Flocchini et al. [8, Section 3] characterized the set of (binary) vectors with the property that restricting input vectors to this set suffices for the solvability of those four relations.

Hirschberg and Sinclair [9] provided the *first* efficient algorithm for Leader Election on asynchronous eponymous rings; that algorithm was based on the intuitive idea of domination in local neighborhoods with progressively doubling size. This algorithm is *uniform*: processors require no externally provided information regarding the ring size, as this can be determined exactly thanks to the assumption of ring eponymity. On rings of size n, the algorithm achieves an $\mathcal{O}(n \cdot \lg n)$ upper bound on message complexity. A corresponding lower bound of $\Omega(n \cdot \lg n)$ has been established in [5]; obviously, this lower bound carries to partially eponymous rings as well.

1.3. Contribution

Throughout, we only consider sets of rings (of the same size n) such that the multiset of identity multiplicities is the same for each ring in any given set. Call it a *set of rings with fixed multiplicities*, or simply a *set of rings*, and denote it as **ID**. So, each ring in a set of rings corresponds to a different arrangement of the identities. All presented results on solvability, computability, and message complexity apply to such sets of rings.

To derive the results, we develop a few preliminary technical notions, which we describe here informally. To measure the circular symmetry in a vector, we use the *period*: the smaller the symmetry, the larger the period. The *(initial) configuration* of a ring consists of the identities and the inputs. The *Minimum Base* of the initial configuration is the period of a composite vector obtained from the initial configuration; so, the Minimum Base measures the circular asymmetry in the initial configuration. It turns out that the Minimum Base and some derivative parameters enjoy elegant number-theoretic expressions, which allow for their efficient evaluation (Section 3.3).

1.3.1. Solvability

We present a topological characterization of solvability for circularly symmetric relations on a set of rings. In more detail, we introduce a new, abstract topological concept, called *compatibility* (Definition 4.1), to capture the possibility that symmetries in the initial configuration persist to the reached outputs; this amounts to demanding that the period of some admissible output vector divides the Minimum Base of the initial configuration. Hence, assuming that the relation is presented explicitly as input to a sequential algorithm (and, therefore, it can be exhaustively searched in time linear in its representation size), compatibility can be checked efficiently since it reduces to checking (a linear number of times) an efficiently checkable number-theoretic predicate. Compatibility can be extended to a pair consisting of a relation and a set of rings in the natural way (Definition 4.2).

We prove that a circularly symmetric relation is solvable on a set of rings if and only if it is compatible with it (Theorem 4.3). Since compatibility can be checked efficiently, this characterization of solvability is *efficient* — in fact, the time complexity of deciding solvability[3] is proportional to the representation size of the given relation.

As an application of this general characterization, we derive a characterization of solvability for the special case of *k-periodic* relations (Theorem 4.6). A characterization of solvability for the particular case of *uniperiodic* and *aperiodic* relations then follows (Corollaries 4.7 and 4.8, respectively). Note that Leader Election is an example of aperiodic relations. Since Leader Election and *all* aperiodic relations share the same characterization of solvability, it follows that the class of aperiodic relations provides a topological characterization of relations that are equivalent to Leader Election with respect to solvability.

[3] We remark that the proof of the characterization invokes only synchronous executions; hence solvability of circularly symmetric relations enjoys identical characterizations in both the synchronous and the asynchronous models, which collapse in this regard. (In fact, the characterization applies to any set of executions that contains the synchronous execution.)

1.3.2. Computability

We present a *universal* distributed algorithm to compute any arbitrary (circularly symmetric) relation on a set of rings (Theorem 4.4). This universal distributed algorithm is comprised of *any* distributed algorithm for constructing the *view* of each processor, followed by local steps; so, this universal algorithm is designed with no concern about message complexity. The distributed algorithm for constructing views is the same for all relations; the local steps are specific to the particular relation. Note that the universal distributed algorithm is non-uniform if and only if the distributed algorithm for constructing views is. Finally, we remark that this universal distributed algorithm (and all later distributed algorithms that will invoke it) makes no assumption on synchrony; hence it works correctly no matter what subset of executions one allows.

1.3.3. Message complexity

We introduce *Multiple Leader Election*, a natural generalization of Leader Election in which a (non-zero) number of leaders must be elected; the number is constrained by some arbitrary function.

We then present a distributed algorithm $\mathcal{A}_{\mathrm{MLE}}(\alpha)$, which satisfies a correctness property reminiscent of Multiple Leader Election; specifically, it leads a certain number of processors to terminate as leaders (Proposition 5.5), and the number depends on the initial configuration. The algorithm runs advised with a lower bound α on the Minimum Base for *any* particular configuration. The advice is "hard-wired" into the common local algorithm of all processors, much in the same way the ring size is in a (standard) non-uniform distributed algorithm.

This distributed algorithm exploits the idea of doubling neighborhoods from the distributed algorithm of Hirschberg and Sinclair [9] for solving Leader Election on eponymous rings. Roughly speaking, processors compare prefixes of views to figure out if they should enter the next *phase*, where the neighborhood size will be doubled. The number of elected leaders depends on the advice α (Proposition 5.5); furthermore, the algorithm achieves message complexity $\mathcal{O}(n \cdot \lg \alpha)$ for any advice α (Proposition 5.4).

In turn, we use the distributed algorithm associated with Multiple Leader Election as the chief building block inside a *universal* distributed algorithm to compute an arbitrary (circularly symmetric) relation Ψ on a set of rings **ID**. This universal distributed algorithm is a particular instantiation of the universal distributed algorithm described in Section 1.3.2; the instantiation is designed with the goal of achieving low message complexity. The resulting distributed algorithm is non-uniform, since it employs a non-uniform distributed algorithm for constructing views.

The obtained algorithm has message complexity $\mathcal{O}\left(\frac{n^2}{\mathsf{LMB}(\mathbf{ID},\Psi)} + n \cdot \lg \mathsf{LMB}(\mathbf{ID},\Psi)\right)$, where $\mathsf{LMB}(\mathbf{ID},\Psi)$ is the *Least Minimum Base* — the least value of Minimum Base over all configurations with an identity vector from **ID** and an input vector from the domain of Ψ (Theorem 5.6). Here, $\mathsf{LMB}(\mathbf{ID},\Psi)$ is used as the advice α for the distributed algorithm associated with Multiple Leader Election.[4] Interestingly, the established upper bound demonstrates that the message complexity on a set of rings **ID** degrades gracefully with the Least Minimum Base (which indicates the degree of least possible eponymity for **ID**); it ranges from $\mathcal{O}(n \cdot \lg n)$ for sets of eponymous rings to $\mathcal{O}(n^2)$ for sets of anonymous rings. We remark that the universal upper bound is *tight* for these two extreme models since $\mathcal{O}(n \cdot \lg n)$ and $\mathcal{O}(n^2)$ are correspondingly tight for some particular relations [2,5].[5]

We are finally interested in determining sets of rings on which the universal upper bound on message complexity from Theorem 5.6 is as low as possible. More specifically, on which sets of rings is it possible to achieve the upper bound of $\mathcal{O}(n \cdot \lg n)$ for *every* relation? (Recall that $\mathcal{O}(n \cdot \lg n)$ is *optimal* for eponymous rings [5].) We identify two such classes of sets of rings:

- Say that a set of rings is *universal* if Leader Election is solvable on it; so, every relation is solvable on a universal set. We prove that a (circularly symmetric) relation is computable with $\mathcal{O}(n \cdot \lg n)$ messages on a universal set of rings (Theorem 5.8). Hence, surprisingly, Leader Election is either unsolvable on a given set of rings, or efficiently computable on the given set with $\mathcal{O}(n \cdot \lg n)$ messages. This efficient bound implies that the message complexity can indeed become smaller if one considers only networks on which a relation is solvable.
- Say that a set of rings is *logarithmic* if each identity appears at most $\mathcal{O}(\lg n)$ times. We prove that each (circularly symmetric) relation is computable with $\mathcal{O}(n \cdot \lg n)$ messages on a logarithmic set of rings (Corollary 5.11). This follows from a more general result we prove about μ-*bounded* sets of rings, where each identity appears at most μ times (Theorem 5.10). Note that this efficient bound on message complexity (for computability) holds even if the relation is *not* solvable on the given set of rings.

We remark that the two particular classes of sets of rings identified in Theorems 5.8 and 5.10, namely universal and μ-bounded, are *incomparable* to each other.[6]

[4] Note that such advice is permissible when designing a distributed algorithm to compute the relation Ψ on the set of rings **ID**, since it provides no information specific to the particular ring on which the distributed algorithm is running each time; the advice provides information pertaining *only* to the membership of the ring in **ID**.

[5] $\mathcal{O}(n \cdot \lg n)$ is optimal for Leader Election on eponymous rings (assuming *uniform* distributed algorithms) [5]; $\mathcal{O}(n^2)$ is optimal for the Sum function and certain other agreement-like relations on anonymous rings [2].

[6] For instance, a set of rings where each identity has multiplicity 2 is clearly 2-bounded (and hence logarithmic), but it is *not* universal. To see this, observe that the set contains a ring of (even) size n where the identities are arranged so that the two copies of each identity appear at distance $\frac{n}{2}$ from

1.4. Comparison

Computability was treated before in [7,8]; however, no explicit formal definition of computability was provided there.

- Dobrev and Pelc [7, Section 1.2] defined the so-called *Generalized Leader Election* problem with the following requirements for the election of one leader: (1) Processors must be able to decide that leader election is *possible*; inputs for which this is possible were called *ambiguous*. (2) Processors must decide that leader election *can be performed*. The statements of both requirements are informal.
- In each of the four relations they considered, Flocchini *et al.* [8, Section 2.2] required that a leader be elected *if possible*; no formal interpretation of this possibility was provided.

To the best of our knowledge, our work is the *first* to provide a formal and crisp definition of computability and attempt a formal distinction between solvability and computability.

Minimum Base was originally defined in [4,10] in terms of graph coverings and graph fibrations; it was used in [3,6] to obtain characterizations of solvability on anonymous networks. In contrast, we exploit the very simple structure of the ring in order to derive a particularly simple version of Minimum Base for partially eponymous rings. For the case of the anonymous ring, Attiya et al. [2] defined the *Symmetry Index* to measure the symmetry in an initial configuration (containing only inputs); in contrast, (Least) Minimum Base measures asymmetry while taking both inputs and identities into account.

Boldi and Vigna [3] provided an *effective* characterization of solvability for any arbitrary relation on an arbitrary network (with any degree of anonymity); in contrast, this work provides the *first efficient* characterization of solvability for any arbitrary (circularly symmetric) relation on partially eponymous rings. It is not evident how the effective graph-theoretic characterization from [3] could specialize to yield an efficient one for the special case of partially eponymous rings. In fact, our goal has been to derive a *direct* characterization of solvability for the particular case of partially eponymous rings that bypasses the complex graph-theoretic framework developed in [3] for the general case. Although the work in [3] invested a great effort in translating concepts of Distributed Computing into some complex graph-theoretic form, our proof techniques are elementary.

The major difference between the proposed algorithm $\mathcal{A}_{\mathsf{MLE}}(\alpha)$ and the algorithm of Hirschberg and Sinclair [9] is that $\mathcal{A}_{\mathsf{MLE}}(\alpha)$ awards processors to proceed to the next phase on the basis of constructed prefixes of processors' views as opposed to processors' identities. Since the *lexicographic ordering* ensures that views and their corresponding prefixes are consistently ordered, the comparison of prefixes by processors in $\mathcal{A}_{\mathsf{MLE}}(\alpha)$ is essentially a comparison of views; it is an efficient one since it avoids the full construction of all views. This is an essential feature of the algorithm $\mathcal{A}_{\mathsf{MLE}}(\alpha)$, and we view its achieved message efficiency as being due to this feature.

Theorem 5.6 improves [8, Theorem 4.2] in three fronts. First, it applies to rings of an *arbitrary* size n, while [8, Theorem 4.2] assumed that n is prime. Second, [8, Theorem 4.2] assumed binary inputs, while Theorem 5.6 makes no assumption on either inputs or identities. (We stress that the assumption of binary inputs was responsible for the $n \cdot \lg n$ term in the upper bound of [8, Theorem 4.2]; in contrast, the $n \cdot \lg \mathsf{LMB}(\mathbf{ID}, \Psi) = \mathcal{O}(n \cdot \lg n)$ term in the upper bound of Theorem 5.6 is due to the application of the neighborhood doubling technique from [9].) Third, and most important, Theorem 5.6 applies to *any* arbitrary (circularly symmetric) relation, while [8, Theorem 4.2] is tailored to four specific relations (Vertex Election, Edge Election, General Election and Multiset Sorting). We remark, however, that the *worst-case* message complexity in both Theorem 5.6 and [8, Theorem 4.2] is $\Theta\left(n^2\right)$. Note also that [8, Theorem 8] is the special case of Corollary 4.8 where Ψ is the General Election relation.

Dobrev and Pelc [7, Theorem 3.1] proved an $\Omega(M \cdot n)$ lower bound on message complexity for the computability of Leader Election on partially eponymous rings, where M is an upper bound on the ring size n known to the processors; this implies a corresponding $\Omega(n^2)$ lower bound when the ring size is known exactly. This lower bound applies to the set of *all* rings of size n; hence, it does not contradict the upper bound in Theorem 5.6, which only applies to a set of rings $\mathbf{ID}$.

Chalopin et al. [6] independently considered a generalization of Leader Election, called *k-Grouping*, which is similar to Multiple Leader Election. k-Grouping requires that processors in a network N, with size $|N|$ a multiple of k, partition themselves into $\frac{|N|}{k}$ groups, each of size exactly k; in contrast, Multiple Leader Election prescribes an upper bound on the number of processors to be elected as leaders. Since any two elected leaders determine a group (consisting of the clockwise left-most leader and all processors between the two leaders), Multiple Leader Election implicitly prescribes an upper bound on the number of groups to be formed (with no fixed number of processors per group). In this sense, k-Grouping imposes some more stringent requirement than Multiple Leader Election. Chalopin et al. [6, Section 3] provide necessary and sufficient conditions for the solvability of k-Grouping under several assumptions on available information about the network (e.g., its size) by the processors.

The motivation for considering sets of rings where each identity appears only a small number of times comes directly from the work of Yamashita and Kameda [15, Section 8]:

each other. The symmetry induced by this arrangement of identities makes the election of a single leader impossible in a synchronous execution (cf. [1, Theorem 4.2]).

> "The anonymous network can be considered as a model of the worst case where the uniqueness [of processor identity numbers] is completely violated. In most practical situations, only a small number of processors may have the same identity number as a result of some failure or design error, and algorithms for solving different problems could take advantage of the (non-unique) identity numbers."

1.5. Road map

Some mathematical preliminaries are articulated in Section 2. The model of a partially eponymous ring is outlined in Section 3. Section 4 treats solvability and computability, while message complexity is treated in Section 5. We conclude, in Section 6, with some open problems.

2. Mathematical preliminaries

2.1. Notation

Denote as $\mathbb{N} = \{0, 1, 2, \ldots\}$, $\mathbb{Z}^+ = \{1, 2, 3, \ldots\}$, and $[n] = \{0, 1, \ldots, n-1\}$ for each integer $n \geq 1$. Denote as GCD and LCM the functions mapping a (multi)set of integers to their *Greatest Common Divisor* and *Least Common Multiple*, respectively. Throughout, fix some integer $n \geq 2$.

2.2. Vectors and equivalence classes

We assume a (possibly infinite) ground set Σ (containing 0 and 1), on which we make no a priori assumptions.[7] We consider a *vector* $\mathbf{x} = \langle x_0, x_1, \ldots, x_{n-1}\rangle \in \Sigma^n$. We use λ to denote the empty vector, while $\mathbf{x} \diamond \mathbf{y}$ denotes the (vector) concatenation of vectors $\mathbf{x}$ and $\mathbf{y}$. With each vector $\mathbf{x}$, we associate a multiset $\mathsf{M}(\mathbf{x})$ with the *multiplicities* of the entries of $\mathbf{x}$; so, the sum of multiplicities is n. We use the function M to partition Σ^n into *equivalence classes*, where all vectors in an equivalence class have the same image under M. Denote as $\mathbf{X}$ the equivalence class containing the vector $\mathbf{x}$. By abuse of notation, $\mathsf{M}(\mathbf{X})$ will denote $\mathsf{M}(\mathbf{x})$ for any vector $\mathbf{x} \in \mathbf{X}$. Note that $\mathsf{GCD}(\mathsf{M}(\mathbf{X}))$ divides each multiplicity in vectors of $\mathbf{X}$. Hence, $\mathsf{GCD}(\mathsf{M}(\mathbf{X}))$ divides n as well.

Fix now an integer $k \in [n]$ and a vector $\mathbf{x} \in \Sigma^n$. The ***(cyclic) shift*** $\sigma_k(\mathbf{x})$ of vector $\mathbf{x}$ is defined as the vector $\langle x_k, x_{k+1}, \ldots, x_{k+n-1}\rangle$, with indices taken modulo n; so, σ_k shifts $\mathbf{x}$ k places anti-clockwise. Note that $\sigma_n(\mathbf{x}) = \mathbf{x}$. The definition is extended to all (including negative) integers k in the natural way.

2.3. Prefixes

For each integer $k \in [n+1]$, the ***prefix of order*** k of $\mathbf{x}$, denoted as $\pi_k(\mathbf{x})$, is given by $\pi_k(\mathbf{x}) = \langle x_0, x_1, \ldots, x_{k-1}\rangle$, with $\pi_0(\mathbf{x}) = \lambda$. Observe that for every vector $\mathbf{x} \in \Sigma^n$ and integer $k \in [n+1]$, $\pi_k(\mathbf{x}) = \pi_k(\mathbf{x} \diamond \mathbf{y})$ for any vector $\mathbf{y}$.

Lemma 2.1. *Consider a vector $\mathbf{x} \in \Sigma^n$ and integers $k, \ell, m \in [n+1]$ such that $k \leq m - \ell$. Then, $\pi_k(\sigma_\ell(\pi_m(\mathbf{x}))) = \pi_k(\sigma_\ell(\mathbf{x}))$.*

Proof. Write $\mathbf{x} = \mathbf{x}_1 \diamond \mathbf{x}_2 \diamond \mathbf{x}_3$, where $\mathbf{x}_1 \in \Sigma^\ell$, $\mathbf{x}_2 \in \Sigma^{m-\ell}$, and $\mathbf{x}_3 \in \Sigma^{n-m}$. Then,

$$\begin{aligned}
\pi_k(\sigma_\ell(\pi_m(\mathbf{x}))) &= \pi_k(\sigma_\ell(\pi_m(\mathbf{x}_1 \diamond \mathbf{x}_2 \diamond \mathbf{x}_3)))\\
&= \pi_k(\sigma_\ell(\mathbf{x}_1 \diamond \mathbf{x}_2))\\
&= \pi_k(\mathbf{x}_2 \diamond \mathbf{x}_1)\\
&= \pi_k(\mathbf{x}_2)\\
&= \pi_k(\mathbf{x}_2 \diamond \mathbf{x}_3 \diamond \mathbf{x}_1)\\
&= \pi_k(\sigma_\ell(\mathbf{x}_1 \diamond \mathbf{x}_2 \diamond \mathbf{x}_3))\\
&= \pi_k(\sigma_\ell(\mathbf{x})),
\end{aligned}$$

as needed. ■

2.4. Periods of vectors

The ***period*** $\mathsf{T}(\mathbf{x})$ of vector $\mathbf{x}$ is the *least* integer k, $0 < k \leq n$, such that $\sigma_k(\mathbf{x}) = \mathbf{x}$. So, $\sigma_{\mathsf{T}(\mathbf{x})}(\mathbf{x}) = \mathbf{x}$, and for every integer $m \in \mathbb{N}$, $\sigma_m(\mathbf{x}) = \sigma_{m \bmod \mathsf{T}(\mathbf{x})}(\mathbf{x})$. Intuitively, the period captures the degree of circular asymmetry of a vector: the smaller

[7] In fact, depending on the sets of rings one considers, Σ could be of some finite size (e.g., for anonymous rings), or should be necessarily infinite (e.g., for eponymous rings).

the period, the more circular symmetries the vector has. Say that $\mathbf{x}$ is $\mathsf{T}(\mathbf{x})$-***periodic***; $\mathbf{x}$ is ***aperiodic*** if $\mathsf{T}(\mathbf{x}) = n$, and $\mathbf{x}$ is ***uniperiodic*** if $\mathsf{T}(\mathbf{x}) = 1$. In general, $\mathbf{x}$ is k-***periodic*** if $\mathsf{T}(\mathbf{x}) = k$.

Say that $\mathbf{x}$ is ***eponymous*** if each entry of $\mathbf{x}$ is unique; clearly, an eponymous vector is aperiodic, but not vice versa. Say that $\mathbf{x}$ is ***anonymous*** if all entries of $\mathbf{x}$ are identical; so, a vector is anonymous if and only if it is uniperiodic. It is simple to show that the period of a vector is invariant under cyclic shifts.

Lemma 2.2. *Consider a vector $\mathbf{x} \in \Sigma^n$. Then, $\mathsf{T}(\mathbf{x}) = \mathsf{T}(\sigma_k(\mathbf{x}))$ for every integer $k \in \mathbb{N}$.*

Proof. Assume, by way of contradiction, that $\mathsf{T}(\mathbf{x}) \neq \mathsf{T}(\sigma_k(\mathbf{x}))$ for some integer $k \in \mathbb{N}$. Without loss of generality, take that $\mathsf{T}(\mathbf{x}) < \mathsf{T}(\sigma_k(\mathbf{x}))$. Since $\sigma_{\mathsf{T}(\mathbf{x})}(\mathbf{x}) = \mathbf{x}$, $\sigma_k(\sigma_{\mathsf{T}(\mathbf{x})}(\mathbf{x})) = \sigma_k(\mathbf{x})$, or $\sigma_{\mathsf{T}(\mathbf{x})}(\sigma_k(\mathbf{x})) = \sigma_k(\mathbf{x})$. This implies that $\mathsf{T}(\sigma_k(\mathbf{x})) \leq \mathsf{T}(\mathbf{x})$. A contradiction. ■

We continue to prove:

Lemma 2.3. *For each vector $\mathbf{x} \in \Sigma^n$ and a pair of integers $\ell, m \in \mathbb{N}$, $\sigma_\ell(\mathbf{x}) = \sigma_m(\mathbf{x})$ if and only if $\ell \equiv m \pmod{\mathsf{T}(\mathbf{x})}$.*

Proof. Recall that $\sigma_{\ell-m}(\mathbf{x}) = \sigma_{(\ell-m) \bmod \mathsf{T}(\mathbf{x})}(\mathbf{x})$. Assume first that $\sigma_\ell(\mathbf{x}) = \sigma_m(\mathbf{x})$. Then,

$$\begin{aligned}
\sigma_{(\ell-m) \bmod \mathsf{T}(\mathbf{x})}(\mathbf{x}) &= \sigma_{\ell-m}(\mathbf{x}) \\
&= \sigma_{-m}(\sigma_\ell(\mathbf{x})) \\
&= \sigma_{-m}(\sigma_m(\mathbf{x})) && \text{(by assumption)} \\
&= \mathbf{x}.
\end{aligned}$$

It follows that $(\ell - m) \bmod \mathsf{T}(\mathbf{x})$ is a (possibly zero) multiple of $\mathsf{T}(\mathbf{x})$. Since $(\ell - m) \bmod \mathsf{T}(\mathbf{x}) < \mathsf{T}(\mathbf{x})$, it follows that $(\ell-m) \bmod \mathsf{T}(\mathbf{x}) = 0$. Hence, $\ell \equiv m \pmod{\mathsf{T}(\mathbf{x})}$, which establishes the first direction. Assume now that $\ell \equiv m \pmod{\mathsf{T}(\mathbf{x})}$. Then,

$$\begin{aligned}
\sigma_\ell(\mathbf{x}) &= \sigma_m(\sigma_{\ell-m}(\mathbf{x})) \\
&= \sigma_m(\sigma_{(\ell-m) \bmod \mathsf{T}(\mathbf{x})}(\mathbf{x})) \\
&= \sigma_m(\sigma_0(\mathbf{x})) && \text{(by assumption)} \\
&= \sigma_m(\mathbf{x}),
\end{aligned}$$

which proves the second direction and completes the proof. ■

We finally prove some number-theoretic properties of period:

Lemma 2.4. *Consider a vector $\mathbf{x} \in \Sigma^n$. Then, (1) $\mathsf{T}(\mathbf{x})$ divides n, and (2) $\frac{n}{\mathsf{GCD}(\mathsf{M}(\mathbf{X}))}$ divides $\mathsf{T}(\mathbf{x})$.*

Proof. For Condition (1), note that $\sigma_n(\mathbf{x}) = \sigma_0(\mathbf{x})$. From Lemma 2.3, it follows that $n \equiv 0 \pmod{\mathsf{T}(\mathbf{x})}$. Thus, $\mathsf{T}(\mathbf{x})$ divides n, as needed.

For Condition (2), note that every entry of $\mathbf{x}$ appears a multiple of $\frac{n}{\mathsf{T}(\mathbf{x})}$ times in $\mathbf{x}$. Hence, the multiplicity of every entry of $\mathbf{x}$ is divided by $\frac{n}{\mathsf{T}(\mathbf{x})}$. Recall that $\mathsf{M}(\mathbf{X})$ is the multiset of multiplicities of the elements in $\mathbf{x} \in \mathbf{X}$. Thus, every multiplicity in the multiset $\mathsf{M}(\mathbf{X})$ is divided by $\frac{n}{\mathsf{T}(\mathbf{x})}$. Recall that a common divisor of a (multi)set of numbers is also a divisor of the Greatest Common Divisor of the (multi)set of numbers. It follows that the common divisor $\frac{n}{\mathsf{T}(\mathbf{x})}$ of the multiset $\mathsf{M}(\mathbf{X})$ is also a divisor of $\mathsf{GCD}(\mathsf{M}(\mathbf{X}))$. Since $\mathsf{GCD}(\mathsf{M}(\mathbf{X}))$ divides n, this implies that $\frac{n}{\mathsf{GCD}(\mathsf{M}(\mathbf{X}))}$ divides $\mathsf{T}(\mathbf{x})$, as needed. ■

2.5. Min-period vectors

Call a vector $\widetilde{\mathbf{x}} \in \mathbf{X}$ a ***min-period*** vector of $\mathbf{X}$ if it achieves the least period among all vectors in $\mathbf{X}$. Lemma 2.4 (Condition (2)) implies that a vector $\mathbf{x} \in \mathbf{X}$ with $\mathsf{T}(\mathbf{x}) = \frac{n}{\mathsf{GCD}(\mathsf{M}(\mathbf{X}))}$ is a min-period vector of $\mathbf{X}$. We prove:

Lemma 2.5. *For each equivalence class $\mathbf{X} \subseteq \Sigma^n$, (1) $\mathsf{T}(\widetilde{\mathbf{x}}) = \frac{n}{\mathsf{GCD}(\mathsf{M}(\mathbf{X}))}$, and (2) for each vector $\mathbf{x} \in \mathbf{X}$, $\mathsf{T}(\widetilde{\mathbf{x}})$ divides $\mathsf{T}(\mathbf{x})$.*

Proof. For Condition (1), it suffices to identify a min-period vector of $\mathbf{X}$ with period $\frac{n}{\mathsf{GCD}(\mathsf{M}(\mathbf{X}))}$. Construct from $\mathsf{M}(\mathbf{X})$ the multiset $\frac{1}{\mathsf{GCD}(\mathsf{M}(\mathbf{X}))}\mathsf{M}(\mathbf{X})$; that is, each entry in $\mathsf{M}(\mathbf{X})$ is divided by $\mathsf{GCD}(\mathsf{M}(\mathbf{X}))$. Fix any arbitrary vector $\mathbf{x}$ such that $\mathsf{M}(\mathbf{x}) = \frac{1}{\mathsf{GCD}(\mathsf{M}(\mathbf{X}))}\mathsf{M}(\mathbf{X})$; so, the multiplicity of each element in $\mathbf{x}$ equals a corresponding multiplicity from $\mathsf{M}(\mathbf{X})$ divided by $\mathsf{GCD}(\mathsf{M}(\mathbf{X}))$. Consider the vector $\mathbf{x} \diamond \cdots \diamond \mathbf{x}$ obtained by concatenating $\mathbf{x}$ with itself $\mathsf{GCD}(\mathsf{M}(\mathbf{X}))$ times. Note that by construction, $\mathsf{T}(\mathbf{x}\diamond\cdots\diamond\mathbf{x}) \leq \frac{n}{\mathsf{GCD}(\mathsf{M}(\mathbf{X}))}$. Clearly, $\mathsf{M}(\mathbf{x}\diamond\ldots\diamond\mathbf{x}) = \mathsf{M}(\mathbf{X})$ and $\mathbf{x}\diamond\cdots\diamond\mathbf{x} \in \mathbf{X}$. Since $\mathbf{x}\diamond\cdots\diamond\mathbf{x} \in \Sigma^n$, Lemma 2.4 (Condition (2)) implies that $\frac{n}{\mathsf{GCD}(\mathsf{M}(\mathbf{X}))}$ divides $\mathsf{T}(\mathbf{x} \diamond \ldots \diamond \mathbf{x})$. It follows that $\mathsf{T}(\mathbf{x} \diamond \cdots \diamond \mathbf{x}) = \frac{n}{\mathsf{GCD}(\mathsf{M}(\mathbf{X}))}$, so that $\mathbf{x} \diamond \ldots \diamond \mathbf{x}$ is a min-period vector of $\mathbf{X}$. Condition (1) now follows.

Condition (2) now follows from Condition (1) and Lemma 2.4 (Condition (2)). ■

2.6. Periods of equivalence classes

Say that the equivalence class $\mathbf{X}$ is ***aperiodic*** if each vector $\mathbf{x} \in \mathbf{X}$ is aperiodic; say that $\mathbf{X}$ is ***uniperiodic*** if each vector $\mathbf{x} \in \mathbf{X}$ is uniperiodic. Say that $\mathbf{X}$ is ***k-periodic*** if a min-period vector $\widetilde{\mathbf{x}}$ of $\mathbf{X}$ is k-periodic. So, aperiodic and uniperiodic are synonyms for n-periodic and 1-periodic, respectively.[8] Lemma 2.5 (Condition (1)) implies that $\mathbf{X}$ is $\frac{n}{\mathsf{GCD}(\mathsf{M}(\mathbf{X}))}$-periodic. Hence, $\mathbf{X}$ is aperiodic if and only if $\mathsf{GCD}(\mathsf{M}(\mathbf{X})) = 1$; $\mathbf{X}$ is uniperiodic if and only if $\mathsf{GCD}(\mathsf{M}(\mathbf{X})) = n$. Clearly, every equivalence class $\mathbf{X}$ is k-periodic for some particular choice of k.

Say that $\mathbf{X}$ is ***anonymous*** if each vector $\mathbf{x} \in \mathbf{X}$ is anonymous; say that $\mathbf{X}$ is ***eponymous*** if each vector $\mathbf{x} \in \mathbf{X}$ is eponymous. It follows that $\mathbf{X}$ is anonymous if and only if it is uniperiodic; $\mathbf{X}$ is eponymous only if it is aperiodic.

2.7. Shift-minimal vectors

We use the standard (*total*) *lexicographic ordering* $\preceq$ on Σ^n. We write $\mathbf{x} \prec \mathbf{y}$ to denote that $\mathbf{x} \preceq \mathbf{y}$ and $\mathbf{x} \neq \mathbf{y}$. Clearly, for all pairs of integers $k < \ell$, $\pi_k(\mathbf{x}) \prec \pi_k(\mathbf{y})$ implies that $\pi_\ell(\mathbf{x}) \prec \pi_\ell(\mathbf{y})$ (and, in particular, $\mathbf{x} \prec \mathbf{y}$). Say that $\mathbf{x} \in \Sigma^n$ is ***shift-minimal*** if it is minimal (with respect to $\preceq$) among all shifts of it; hence, $\mathbf{x}$ is shift-minimal if $\mathbf{x} \preceq \sigma_k(\mathbf{x})$ for all integers $k \in [n]$. We prove a simple property of shift-minimal vectors:

Lemma 2.6. *Consider a shift-minimal vector $\mathbf{x} \in \Sigma^n$, and a pair of integers $\ell, m \in [n]$. Then,* (1) *$\pi_\ell(\mathbf{x}) \preceq \pi_\ell(\sigma_m(\mathbf{x}))$, and* (2) *if $m \not\equiv 0 \pmod{\mathsf{T}(\mathbf{x})}$ and $\ell \geq m$, then $\pi_\ell(\mathbf{x}) \prec \pi_\ell(\sigma_{-m}(\mathbf{x}))$.*

Proof. For Condition (1), assume, by way of contradiction, that there is a pair of integers $\ell, m \in [n]$ such that $\pi_\ell(\mathbf{x}) \not\preceq \pi_\ell(\sigma_m(\mathbf{x}))$. Then, $\pi_\ell(\sigma_m(\mathbf{x})) \prec \pi_\ell(\mathbf{x})$. By lexicographic ordering, this implies that $\sigma_m(\mathbf{x}) \prec \mathbf{x}$, so that $\mathbf{x}$ is not shift-minimal. A contradiction.

For Condition (2), assume, by way of contradiction, that there is a pair of integers $\ell, m \in [n]$ such that $m \not\equiv 0 \pmod{\mathsf{T}(\mathbf{x})}$, $\ell \geq m$, and yet $\pi_\ell(\mathbf{x}) \not\prec \pi_\ell(\sigma_{-m}(\mathbf{x}))$. Since $\mathsf{T}(\mathbf{x})$ divides n (by Lemma 2.4 (Condition (1))), it follows that $n - m \not\equiv 0 \pmod{\mathsf{T}(\mathbf{x})}$. Hence, Lemma 2.3 implies that $\sigma_{n-m}(\mathbf{x}) \neq \sigma_0(\mathbf{x}) = \mathbf{x}$.

By Condition (1), $\pi_\ell(\mathbf{x}) \preceq \pi_\ell(\sigma_{n-m}(\mathbf{x}))$. Since $\sigma_{n-m}(\mathbf{x}) = \sigma_{-m}(\mathbf{x})$, the assumption implies that $\pi_\ell(\mathbf{x}) \not\prec \pi_\ell(\sigma_{n-m}(\mathbf{x}))$. It follows that $\pi_\ell(\mathbf{x}) = \pi_\ell(\sigma_{n-m}(\mathbf{x}))$. Since $\ell \geq m$, this implies that $\pi_m(\mathbf{x}) = \pi_m(\sigma_{n-m}(\mathbf{x}))$.

Since $\mathbf{x}$ is shift-minimal, $\mathbf{x} \preceq \sigma_{n-m}(\mathbf{x})$. Since $\mathbf{x} \neq \sigma_{n-m}(\mathbf{x})$, it follows that $\mathbf{x} \prec \sigma_{n-m}(\mathbf{x})$. Since $\pi_m(\mathbf{x}) = \pi_m(\sigma_{n-m}(\mathbf{x}))$, this implies that $\sigma_m(\mathbf{x}) \prec \sigma_m(\sigma_{n-m}(\mathbf{x})) = \sigma_n(\mathbf{x}) = \mathbf{x}$, so that $\mathbf{x}$ is not shift-minimal. A contradiction. ■

2.8. Shuffle vectors

The ***shuffle*** of two vectors $\mathbf{x} = \langle x_0, x_1, \ldots, x_{n-1}\rangle$ and $\mathbf{y} = \langle y_0, y_1, \ldots, y_{n-1}\rangle$, denoted as $\mathbf{x} \parallel \mathbf{y}$, is the vector $\mathbf{x} \parallel \mathbf{y} = \langle (x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\rangle$. Clearly, for any integer $k \in \mathbb{N}$, $\sigma_k(\mathbf{x} \parallel \mathbf{y}) = \sigma_k(x) \parallel \sigma_k(y)$; so, the shift operator and the shuffle operator commute. We prove:

Lemma 2.7. *For each pair of vectors $\mathbf{x}, \mathbf{y} \in \Sigma^n$, $\mathsf{T}(\mathbf{x} \parallel \mathbf{y}) = \mathsf{LCM}(\mathsf{T}(\mathbf{x}), \mathsf{T}(\mathbf{y}))$.*

Proof. Clearly,

$$\sigma_{\mathsf{LCM}(\mathsf{T}(\mathbf{x}),\mathsf{T}(\mathbf{y}))}(\mathbf{x} \parallel \mathbf{y}) = \sigma_{\mathsf{LCM}(\mathsf{T}(\mathbf{x}),\mathsf{T}(\mathbf{y}))}(\mathbf{x}) \parallel \sigma_{\mathsf{LCM}(\mathsf{T}(\mathbf{x}),\mathsf{T}(\mathbf{y}))}(\mathbf{y}).$$

Since, $\mathsf{LCM}(\mathsf{T}(\mathbf{x}), \mathsf{T}(\mathbf{y})) \equiv 0 \pmod{\mathsf{T}(\mathbf{x})}$ and $\mathsf{LCM}(\mathsf{T}(\mathbf{x}), \mathsf{T}(\mathbf{y})) \equiv 0 \pmod{\mathsf{T}(\mathbf{y})}$, Lemma 2.3 implies that $\sigma_{\mathsf{LCM}(\mathsf{T}(\mathbf{x}),\mathsf{T}(\mathbf{y}))}(\mathbf{x}) = \sigma_0(\mathbf{x}) = \mathbf{x}$ and $\sigma_{\mathsf{LCM}(\mathsf{T}(\mathbf{x}),\mathsf{T}(\mathbf{y}))}(\mathbf{y}) = \sigma_0(\mathbf{y}) = \mathbf{y}$. It follows that

$$\sigma_{\mathsf{LCM}(\mathsf{T}(\mathbf{x}),\mathsf{T}(\mathbf{y}))}(\mathbf{x} \parallel \mathbf{y}) = \mathbf{x} \parallel \mathbf{y}.$$

By Lemma 2.3, this implies that $\mathsf{T}(\mathbf{x} \parallel \mathbf{y})$ divides $\mathsf{LCM}(\mathsf{T}(\mathbf{x}), \mathsf{T}(\mathbf{y}))$.

Assume, by way of contradiction, that $\mathsf{LCM}(\mathsf{T}(\mathbf{x}), \mathsf{T}(\mathbf{y}))$ does *not* divide $\mathsf{T}(\mathbf{x} \parallel \mathbf{y})$. It follows that either $\mathsf{T}(\mathbf{x})$ does not divide $\mathsf{T}(\mathbf{x} \parallel \mathbf{y})$ or $\mathsf{T}(\mathbf{y})$ does not divide $\mathsf{T}(\mathbf{x} \parallel \mathbf{y})$. Without loss of generality, take that $\mathsf{T}(\mathbf{x})$ does not divide $\mathsf{T}(\mathbf{x} \parallel \mathbf{y})$. Thus, $\mathsf{T}(\mathbf{x} \parallel \mathbf{y}) \not\equiv 0 \pmod{\mathsf{T}(\mathbf{x})}$. By Lemma 2.3, this implies that $\sigma_{\mathsf{T}(\mathbf{x} \parallel \mathbf{y})}(\mathbf{x}) \neq \sigma_0(\mathbf{x}) = \mathbf{x}$. It follows that

$$\begin{aligned}\sigma_{\mathsf{T}(\mathbf{x} \parallel \mathbf{y})}(\mathbf{x} \parallel \mathbf{y}) &= \sigma_{\mathsf{T}(\mathbf{x} \parallel \mathbf{y})}(\mathbf{x}) \parallel \sigma_{\mathsf{T}(\mathbf{x} \parallel \mathbf{y})}(\mathbf{y}) \\ &\neq \mathbf{x} \parallel \mathbf{y}.\end{aligned}$$

By definition of period, $\sigma_{\mathsf{T}(\mathbf{x} \parallel \mathbf{y})}(\mathbf{x} \parallel \mathbf{y}) = \mathbf{x} \parallel \mathbf{y}$. A contradiction.

Since $\mathsf{LCM}(\mathsf{T}(\mathbf{x}), \mathsf{T}(\mathbf{y}))$ and $\mathsf{T}(\mathbf{x} \parallel \mathbf{y})$ divide each other, the claim follows. ■

[8] For example, $\mathbf{X}$ is 1-periodic if and only if $\mathsf{T}(\widetilde{\mathbf{x}}) = 1$, which is equivalent to $\widetilde{\mathbf{x}}$ being a constant vector. But $\widetilde{\mathbf{x}}$ is a constant vector if and only if every vector $\mathbf{x} \in \mathbf{X}$ is constant (i.e., has period 1). This is equivalent to $\mathbf{X}$ being uniperiodic. So, 1-periodic and uniperiodic equivalence classes coincide.

2.9. Relations

A (recursive) ***relation*** is a (non-empty) subset $\Psi \subseteq \Sigma^n \times \Sigma^n$. For a vector $\mathbf{x} \in \Sigma^n$, define

$$\Psi(\mathbf{x}) = \{\mathbf{y} \mid (\mathbf{x}, \mathbf{y}) \in \Psi\};$$

so, a vector $\mathbf{y} \in \Psi(\mathbf{x})$ is an ***image*** of $\mathbf{x}$ under Ψ. The set $\mathsf{Dom}(\Psi)$ of all vectors $\mathbf{x} \in \Sigma^n$ with at least one image under Ψ is the ***domain*** of Ψ; so, $\mathsf{Dom}(\Psi) = \{\mathbf{x} \in \Sigma^n \mid \Psi(\mathbf{x}) \neq \emptyset\}$. The relation Ψ is ***total*** if $\mathsf{Dom}(\Psi) = \Sigma^n$. The set of all images of all vectors $\mathbf{x} \in \Sigma^n$ is the ***image*** of Ψ, denoted as $\mathsf{Im}(\Psi)$; so, $\mathsf{Im}(\Psi) = \bigcup_{\mathbf{x} \in \Sigma^n} \Psi(\mathbf{x})$.

The relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is ***aperiodic*** if each vector in $\mathsf{Im}(\Psi)$ is aperiodic; so, each image under an aperiodic relation has no circular symmetries. On the other extreme, the relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is ***uniperiodic*** if each vector in $\mathsf{Im}(\Psi)$ is uniperiodic; so, each image under a uniperiodic relation is constant. In the middle range, the relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is ***k-periodic*** if each vector in $\mathsf{Im}(\Psi)$ is k-periodic. So, aperiodic and uniperiodic are synonyms for n-periodic and 1-periodic, respectively. Note that *not* every relation is k-periodic for some particular choice of k since the image of a relation may include vectors with different periods.

Given two relations $\Psi_1, \Psi_2 \subseteq \Sigma^n \times \Sigma^n$, their ***composition*** is the relation

$$\Psi_1 \circ \Psi_2 = \left\{(\mathbf{x}, \mathbf{y}) \mid (\mathbf{x}, \mathbf{z}) \in \Psi_2 \text{ and } (\mathbf{z}, \mathbf{y}) \in \Psi_1 \text{ for some } \mathbf{z} \in \Sigma^n\right\}.$$

For a relation $\Psi \subseteq \Sigma^n \times \Sigma^n$, note that

$$\sigma_1 \circ \Psi = \{(\mathbf{x}, \mathbf{y}) \mid \mathbf{y} = \sigma_1(\mathbf{z}) \text{ for some } \mathbf{z} \in \Psi(\mathbf{x})\};$$

moreover,

$$\Psi \circ \sigma_1 = \{(\mathbf{x}, \mathbf{y}) \mid \mathbf{y} \in \Psi(\mathbf{z}) \text{ where } \mathbf{z} = \sigma_1(\mathbf{x})\}.$$

In particular, for a vector $\mathbf{x} \in \Sigma^n$,

$$\sigma_1 \circ \Psi(\mathbf{x}) = \{\mathbf{y} \mid \mathbf{y} = \sigma_1(\mathbf{z}) \text{ for some } \mathbf{z} \in \Psi(\mathbf{x})\}$$

and

$$\Psi \circ \sigma_1(\mathbf{x}) = \{\mathbf{y} \mid \mathbf{y} \in \Psi(\mathbf{z}) \text{ where } \mathbf{z} = \sigma_1(\mathbf{x})\}.$$

So, $\sigma_1 \circ \Psi$ maps inputs to shifts of their images; $\Psi \circ \sigma_1$ maps inputs to images of their shifts.

The relation Ψ is ***circularly symmetric*** if $\sigma_1 \circ \Psi \subseteq \Psi \circ \sigma_1$. Intuitively, in a circularly symmetric relation, shifts of images are always images of shifts. A direct induction implies that $\sigma_k \circ \Psi \subseteq \Psi \circ \sigma_k$ for any circularly symmetric relation Ψ and for all integers $k \in \mathbb{N}$.

In the ***Leader Election*** relation $LE \subseteq \Sigma^n \times \Sigma^n$, the set of images of each input vector $\mathbf{x}$ is the set of all binary vectors with exactly one 1 and $(n-1)$ 0's; 1 and 0 correspond to *elected* and *non-elected*, respectively. Clearly, Leader Election is total, aperiodic and circularly symmetric.

We now give a generalization of the Leader Election relation. Consider a function $\Phi : \Sigma^n \to \mathbb{Z}^+$. In the ***$\Phi$-Leader Election*** relation $\Phi\text{-}LE \subseteq \Sigma^n \times \Sigma^n$, the set of images of each input vector $\mathbf{x}$ is the set of all binary vectors with the number of 1's ranging from 1 to $\Phi(\mathbf{x})$ (both inclusive). The special case where $\Phi(\mathbf{x}) = 1$ for all vectors $\mathbf{x} \in \Sigma^n$ is precisely Leader Election. A ***Multiple Leader Election*** relation is a Φ-Leader Election relation for some such function Φ.

3. The partially eponymous ring

3.1. General

We start with the standard model of an *asynchronous*, *anonymous ring* as studied, for example, in [2,9]. A ***ring*** of size n is a cyclic arrangement of n identical ***processors*** $0, 1, \ldots, n-1$. The ring is *oriented* and *bidirectional*, and processors have ***identities***. We augment this model so that the identities are neither necessarily all identical nor necessarily unique. Call it a ***partially eponymous ring***. In an ***anonymous ring***, identities are all identical; in an ***eponymous ring***, identities are unique.

Processor j has an identity id_j and receives an input in_j. The ***identity vector*** is $\mathbf{id} = \langle id_0, id_1, \ldots, id_{n-1}\rangle$; the ***input vector*** is $\mathbf{in} = \langle in_0, in_1, \ldots, in_{n-1}\rangle$. Note that for an anonymous ring, $\mathsf{T}(\mathbf{id}) = 1$; for an eponymous ring, $\mathsf{T}(\mathbf{id}) = n$. The ***(initial) configuration*** of the ring is the tuple $\langle \mathbf{id}, \mathbf{in}\rangle$. Each processor seeks to reach an output out_j by running a (local) *algorithm* and communicating with its two neighbors. The ***output vector*** is $\mathbf{out} = \langle out_0, out_1, \ldots, out_{n-1}\rangle$. The ***i-neighborhood*** of a processor j is the set of processors $\{j-i, \ldots, j-1, j, j+1, \ldots j+i\}$.

There is a *single* (local) ***algorithm*** A run by all processors; A is represented as a (possibly infinite) state machine. Each computation step of A at processor j is dependent on the current state of j, the messages currently received at j and the local identity id_j and input in_j. A ***distributed algorithm*** $\mathcal{A}$ is a collection of local algorithms, one for each processor. We restrict attention to ***non-uniform*** distributed algorithms, where the size of the ring is "hard-wired" into the single local algorithm. So, we consider rings of a certain size n. The distributed algorithm $\mathcal{A}$ induces a set of (***asynchronous***) ***executions***. A subset of

the executions of the distributed algorithm $\mathcal{A}$ are ***synchronous***; there, processors proceed in lock-step to receive messages, perform some local computations and send messages out.

Each identity vector $\mathbf{id} \in \Sigma^n$ specifies a single ring; by abuse of notation, denote as $\mathbf{id}$ the specified ring. An equivalence class $\mathbf{ID} \subseteq \Sigma^n$ induces a set of rings, each corresponding to some particular identity vector $\mathbf{id} \in \mathbf{ID}$; by abuse of notation, denote as $\mathbf{ID}$ the induced set. Recall that each ring in the set $\mathbf{ID}$ has the same multiset of identity multiplicities.

3.2. Solvability and computability

Fix a configuration $\langle \mathbf{id}, \mathbf{in} \rangle$. Say that the distributed algorithm $\mathcal{A}$ ***solves the set of output vectors*** $\mathcal{OUT}$ ***on the configuration*** $\langle \mathbf{id}, \mathbf{in} \rangle$ if each execution of $\mathcal{A}$ on the ring $\mathbf{id}$ with input $\mathbf{in}$ results to an output vector $\mathbf{out} \in \mathcal{OUT}$. Say that the set of output vectors $\mathcal{OUT}$ is ***solvable on the configuration*** $\langle \mathbf{id}, \mathbf{in} \rangle$ if there is a distributed algorithm $\mathcal{A}$ that solves $\mathcal{OUT}$ on $\langle \mathbf{id}, \mathbf{in} \rangle$. We are now ready for a significant definition.

Definition 3.1 (*Solvable Relation*). The relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is ***solvable on the set of rings*** $\mathbf{ID} \subseteq \Sigma^n$ if there is a distributed algorithm $\mathcal{A}$ such that for each configuration $\langle \mathbf{id}, \mathbf{in} \rangle \in \mathbf{ID} \times \mathsf{Dom}(\Psi)$, $\mathcal{A}$ solves $\Psi(\mathbf{in})$ on $\langle \mathbf{id}, \mathbf{in} \rangle$.

The distributed algorithm $\mathcal{A}$ ***computes the set of output vectors*** $\mathcal{OUT}$ ***on the configuration*** $\langle \mathbf{id}, \mathbf{in} \rangle$ if the following holds: if $\mathcal{OUT}$ is solvable on the configuration $\langle \mathbf{id}, \mathbf{in} \rangle$, then $\mathcal{A}$ solves $\mathcal{OUT}$ in $\langle \mathbf{id}, \mathbf{in} \rangle$; else $\mathcal{A}$ solves $\{\perp^n\}$ on $\langle \mathbf{id}, \mathbf{in} \rangle$ (where $\perp$ stands for an *impossibility* output).

We now develop the notion of a distributed algorithm working for a set of rings and on the entire domain of the relation Ψ; intuitively, the set of rings and the relation Ψ represent information that is available to the algorithm. Our next two definitions will simultaneously generalize sets of output vectors to relations, and configurations to sets of rings. We start with the first definition.

Definition 3.2 (*Distributed Algorithm Computing a Relation*). The distributed algorithm $\mathcal{A}$ ***computes the relation*** $\Psi \subseteq \Sigma^n \times \Sigma^n$ ***on a set of rings*** $\mathbf{ID} \subseteq \Sigma^n$ ***with*** $g(n)$ ***messages*** if $g(n)$ is the least function such that for each configuration $\langle \mathbf{id}, \mathbf{in} \rangle \in \mathbf{ID} \times \mathsf{Dom}(\Psi)$, $\mathcal{A}$ computes $\Psi(\mathbf{in})$ on the configuration $\langle \mathbf{id}, \mathbf{in} \rangle$ with no more than $g(n)$ messages in any execution. $g(n)$ is the ***message complexity of*** $\mathcal{A}$ ***for computing*** Ψ ***on*** $\mathbf{ID}$.

We conclude with another significant definition:

Definition 3.3 (*Computable Relation*). The relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is ***computable on a set of rings*** $\mathbf{ID} \subseteq \Sigma^n$ ***with*** $g(n)$ ***messages*** if there is a distributed algorithm $\mathcal{A}$ that computes Ψ on $\mathbf{ID}$ with $g(n)$ messages. The ***message complexity for computing*** Ψ ***on*** $\mathbf{ID}$ is the least message complexity of a distributed algorithm for computing Ψ on $\mathbf{ID}$.

Note that solvability of a relation Ψ on a set of rings $\mathbf{ID}$ implies computability of Ψ on $\mathbf{ID}$ (with *some* number of messages). However, we shall soon see that the inverse does not necessarily hold.

3.3. The least minimum base

We start with a significant definition:

Definition 3.4. The ***Minimum Base*** $\mathsf{MB}(\mathbf{id}, \mathbf{in})$ of a configuration $\langle \mathbf{id}, \mathbf{in} \rangle \in \Sigma^n \times \Sigma^n$ is defined as $\mathsf{MB}(\mathbf{id}, \mathbf{in}) = \mathsf{T}(\mathbf{id} \,\|\, \mathbf{in})$.

Note that since $\mathbf{id} \,\|\, \mathbf{in}$ is a vector of length n, $\mathsf{T}(\mathbf{id} \,\|\, \mathbf{in}) \leq n$; thus $\mathsf{MB}(\mathbf{id}, \mathbf{in}) \leq n$. Note also that Lemma 2.7 immediately implies:

Lemma 3.1. *For a configuration* $\langle \mathbf{id}, \mathbf{in} \rangle \in \Sigma^n \times \Sigma^n$, $\mathsf{MB}(\mathbf{id}, \mathbf{in}) = \mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))$.

For a set of rings $\mathbf{ID} \subseteq \Sigma^n$ and an input vector $\mathbf{in} \in \Sigma^n$, the ***Min-Period Minimum Base*** $\mathsf{MB}(\mathbf{ID}, \mathbf{in})$ is defined as $\mathsf{MB}(\mathbf{ID}, \mathbf{in}) = \mathsf{MB}(\widetilde{\mathbf{id}}, \mathbf{in})$, where $\widetilde{\mathbf{id}}$ is a min-period vector of $\mathbf{ID}$. Thus,

$$\begin{aligned} \mathsf{MB}(\mathbf{ID}, \mathbf{in}) &= \mathsf{LCM}(\mathsf{T}(\widetilde{\mathbf{id}}), \mathsf{T}(\mathbf{in})) && \text{(by Lemma 3.1)} \\ &= \mathsf{LCM}\left(\tfrac{n}{\mathsf{GCD}(\mathsf{M}(\mathbf{ID}))}, \mathsf{T}(\mathbf{in})\right) && \text{(by Lemma 2.5 (Condition (1)))}\ . \end{aligned}$$

Note that if $\mathbf{ID}$ is anonymous, then $\mathsf{MB}(\mathbf{ID}, \mathbf{in}) = \mathsf{LCM}(1, \mathsf{T}(\mathbf{in})) = \mathsf{T}(\mathbf{in})$; if $\mathbf{ID}$ is eponymous, then $\mathsf{MB}(\mathbf{ID}, \mathbf{in}) = \mathsf{LCM}(n, \mathsf{T}(\mathbf{in})) \geq n \geq \mathsf{T}(\mathbf{in})$. So, intuitively, the Min-Period Minimum Base is an indicator of the degree of least possible eponymity for a set of rings. We continue to prove:

Lemma 3.2. *For a set of rings* $\mathbf{ID} \subseteq \Sigma^n$ *and an input vector* $\mathbf{in} \in \Sigma^n$, $\mathsf{MB}(\mathbf{ID}, \mathbf{in})$ *divides* $\mathsf{MB}(\mathbf{id}, \mathbf{in})$ *for each ring* $\mathbf{id} \in \mathbf{ID}$.

Proof. By Lemma 2.5 (Condition (2)), $\mathsf{T}(\widetilde{\mathbf{id}})$ divides $\mathsf{T}(\mathbf{id})$ for each ring $\mathbf{id} \in \mathbf{ID}$. This implies that $\mathsf{LCM}(\mathsf{T}(\widetilde{\mathbf{id}}), \mathsf{T}(\mathbf{in}))$ divides $\mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))$ for each ring $\mathbf{id} \in \mathbf{ID}$. Since $\mathsf{MB}(\mathbf{ID}, \mathbf{in}) = \mathsf{LCM}(\mathsf{T}(\widetilde{\mathbf{id}}), \mathsf{T}(\mathbf{in}))$ and $\mathsf{MB}(\mathbf{id}, \mathbf{in}) = \mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))$ (by Lemma 3.1), it follows that $\mathsf{MB}(\mathbf{ID}, \mathbf{in})$ divides $\mathsf{MB}(\mathbf{id}, \mathbf{in})$ for each ring $\mathbf{id} \in \mathbf{ID}$, as needed. ■

We finally define:

Definition 3.5. The ***Least Minimum Base*** $\mathsf{LMB}(\mathbf{ID}, \Psi)$ of a set of rings $\mathbf{ID} \subseteq \Sigma^n$ and a relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is defined as

$$\mathsf{LMB}(\mathbf{ID}, \Psi) = \min\{\mathsf{MB}(\mathbf{id}, \mathbf{in}) \mid \langle \mathbf{id}, \mathbf{in}\rangle \in \mathbf{ID} \times \mathsf{Dom}(\Psi)\}.$$

Since $\mathsf{MB}(\mathbf{id}, \mathbf{in}) \leq n$ for every configuration $\langle \mathbf{id}, \mathbf{in}\rangle$, it follows that $\mathsf{LMB}(\mathbf{ID}, \Psi) \leq n$.

3.4. Views

We conclude with a definition that extends one in [15, Section 3] to a ring where processors receive inputs. Given a configuration $\langle \mathbf{id}, \mathbf{in}\rangle \in \Sigma^n \times \Sigma^n$, the ***view*** of processor j is

$$view_j(\mathbf{id}, \mathbf{in}) = \sigma_j(\mathbf{id} \parallel \mathbf{in}) = \sigma_j(\mathbf{id}) \parallel \sigma_j(\mathbf{in}).$$

Lemmas 2.2 and 2.7 together imply that $\mathsf{T}(view_j(\mathbf{id}, \mathbf{in})) = \mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))$.

Clearly, views of different processors are cyclic shifts of each other. Note that the configuration $\langle \mathbf{id}, \mathbf{in}\rangle$ uniquely determines the set of views $\{view_j(\mathbf{id}, \mathbf{in}) \mid j \in [n]\}$. There is an immediate non-uniform distributed algorithm $\mathcal{A}_{\mathsf{CV}}$, which collects all identities and inputs at each processor, so that each processor can locally construct its own view; the message complexity of $\mathcal{A}_{\mathsf{CV}}$ is $\Theta(n^2)$. It is simple to prove by a symmetry argument (cf. [1, Theorem 4.2]):

Lemma 3.3. *Fix a configuration $\langle \mathbf{id}, \mathbf{in}\rangle \in \Sigma^n \times \Sigma^n$ and two processors $j, k \in [n]$ with $view_j(\mathbf{id}, \mathbf{in}) = view_k(\mathbf{id}, \mathbf{in})$. Then, in a synchronous execution on $\langle \mathbf{id}, \mathbf{in}\rangle$ of a distributed algorithm with output vector $\mathbf{out}$, $out_j = out_k$.*

4. Solvability and computability

4.1. Preliminaries

We start with a definition of compatibility between a set of output vectors and a configuration.

Definition 4.1 (*Compatibility with a Configuration*). The set of output vectors $\mathcal{OUT} \subseteq \Sigma^n$ is ***compatible*** with the configuration $\langle \mathbf{id}, \mathbf{in}\rangle \in \Sigma^n \times \Sigma^n$ if there is an output vector $\mathbf{out} \in \mathcal{OUT}$ such that $\mathsf{T}(\mathbf{out})$ divides $\mathsf{MB}(\mathbf{id}, \mathbf{in})$.

Recall that $\mathsf{MB}(\mathbf{id}, \mathbf{in})$ can be computed efficiently; thus, compatibility of a set of output vectors with a configuration can be checked efficiently as well. Recall also that by Lemma 3.3, symmetry in the initial configuration $\langle \mathbf{id}, \mathbf{in}\rangle$ may carry over to the outputs of the processors in certain executions; moreover, the symmetry in the initial configuration is captured by $\mathsf{MB}(\mathbf{id}, \mathbf{in})$. So, intuitively, if all output vectors in $\mathcal{OUT}$ have symmetries that are inconsistent with the symmetry of the initial configuration, then none of these output vectors can be returned in certain executions. Thus, a set of output vectors is compatible with a configuration if *not all* output vectors are a priori excluded from being returned. This intuition is formalized and established in the next claim.

Proposition 4.1. *Assume that a set of output vectors $\mathcal{OUT} \subseteq \Sigma^n$ is solvable on a configuration $\langle \mathbf{id}, \mathbf{in}\rangle \in \Sigma^n \times \Sigma^n$. Then, $\mathcal{OUT}$ is compatible with $\langle \mathbf{id}, \mathbf{in}\rangle$.*

Proof. By assumption, there is a distributed algorithm $\mathcal{A}$ that solves $\mathcal{OUT}$ on $\langle \mathbf{id}, \mathbf{in}\rangle$. Fix a synchronous execution of $\mathcal{A}$ on $\langle \mathbf{id}, \mathbf{in}\rangle$, and consider the associated output vector $\mathbf{out} \in \mathcal{OUT}$. Fix a processor j. Then,

$$\begin{aligned} view_{j+\mathsf{T}(\mathbf{id} \parallel \mathbf{in})}(\mathbf{id}, \mathbf{in}) &= \sigma_{j+\mathsf{T}(\mathbf{id} \parallel \mathbf{in})}(\mathbf{id} \parallel \mathbf{in}) && \text{(by definition of view)} \\ &= \sigma_j(\mathbf{id} \parallel \mathbf{in}) && \text{(by Lemma 2.3)} \\ &= view_j(\mathbf{id}, \mathbf{in}) && \text{(by definition of view)}. \end{aligned}$$

Hence, Lemma 3.3 implies that $out_j = out_{j+\mathsf{T}(\mathbf{id} \parallel \mathbf{in})}$. Since j was chosen arbitrarily, this implies that $\sigma_{\mathsf{T}(\mathbf{id} \parallel \mathbf{in})}(\mathbf{out}) = \mathbf{out}$. Thus, Lemma 2.3 implies now that $\mathsf{T}(\mathbf{out})$ divides $\mathsf{T}(\mathbf{id} \parallel \mathbf{in})$. Since $\mathsf{MB}(\mathbf{id}, \mathbf{in}) = \mathsf{T}(\mathbf{id} \parallel \mathbf{in})$, it follows that $\mathsf{T}(\mathbf{out})$ divides $\mathsf{MB}(\mathbf{id}, \mathbf{in})$. Hence, $\mathcal{OUT}$ is compatible with $\langle \mathbf{id}, \mathbf{in}\rangle$, as needed. ■

We now present the distributed algorithm $\mathcal{A}_\Psi$ associated with an arbitrary circularly symmetric relation $\Psi \in \Sigma^n \times \Sigma^n$; the algorithm is described in pseudocode in Fig. 1. Note that the distributed algorithm $\mathcal{A}_\Psi$ does *not* specify how the views are constructed in *Step* 2. These can be constructed by invoking, for example, the distributed algorithm $\mathcal{A}_{\mathsf{CV}}$ from Section 3. All remaining steps of the distributed algorithm $\mathcal{A}_\Psi$ are local and only depend on information collected during *Step* 2. Hence, the distributed algorithm $\mathcal{A}_\Psi$ is non-uniform if and only if the invoked distributed algorithm for constructing views is.

Steps 3–6 enable processors to choose an output vector; *Step* 7 enables processor j to output its individual coordinate in this vector. The set $Choices_j$ contains all common candidates for the output vector; in *Step* 6, all processors use a common function (e.g., min) to single out one of the candidates. We prove:

Proposition 4.2. *Fix a relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ and a configuration $\langle \mathbf{id}, \mathbf{in}\rangle \in \Sigma^n \times \Sigma^n$. Then, either $\mathcal{A}_\Psi$ solves $\Psi(\mathbf{in})$ on $\langle \mathbf{id}, \mathbf{in}\rangle$ or $\mathcal{A}_\Psi$ solves $\{\perp^n\}$ on $\langle \mathbf{id}, \mathbf{in}\rangle$.*

$\mathcal{A}_\Psi$: CODE FOR PROCESSOR j WITH IDENTITY id_j AND INPUT in_j

1: Upon receiving message $\langle wake \rangle$ **do**

2: Construct $view_j(\mathbf{id}, \mathbf{in}) = \mathbf{vid}_j \| \mathbf{vin}_j$. $\quad \backslash * \ \mathbf{vid}_j = \sigma_j(\mathbf{id}),\ \mathbf{vin}_j = \sigma_j(\mathbf{in}) \ * \backslash$

3: $Views_j[i] := \langle \sigma_i(\mathbf{vid}_j), \sigma_i(\mathbf{vin}_j) \rangle$ for each $i \in [n]$, where n is the size of $view_j(\mathbf{id}, \mathbf{in})$.

4: $Choices_j := \{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \mid \langle \mathbf{x}, \mathbf{y} \rangle \in Views_j;\ \mathbf{z} \in \Psi(\mathbf{y});\ \mathsf{T}(\mathbf{z})$ divides $\mathsf{MB}(\mathbf{x}, \mathbf{y})\}$.

5: **If** $Choices_j = \emptyset$ **then** $out_j := \perp$ and terminate.

6: (a) $(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}}) := \min\{Choices_j\}$;

(b) $k_j := \min\left\{i \in [n] \mid Views_j[i] = \langle \underline{\mathbf{x}}, \underline{\mathbf{y}} \rangle\right\}$, where n is the size of $view_j(\mathbf{id}, \mathbf{in})$.

7: Set out_j to the first entry of $\sigma_{-k_j}(\underline{\mathbf{z}})$ and terminate.

Fig. 1. Algorithm $\mathcal{A}_\Psi$: code for processor j.

Proof. Since the set of views $\left\{view_j(\mathbf{id}, \mathbf{in}) \mid j \in [n]\right\}$ is uniquely determined, it follows by the algorithm (*Step* 4) that the set $Choices_j$ constructed at processor j is also uniquely determined and common for all processors. We proceed by case analysis.

1. Assume first that $Choices_j = \emptyset$. Then, by the algorithm (*Step* 5), $\mathbf{out} = \perp^n$. So, $\mathcal{A}_\Psi$ solves $\{\perp^n\}$ on $\langle \mathbf{id}, \mathbf{in} \rangle$.
2. Assume now that $Choices_j \neq \emptyset$. We now prove that the commonly chosen triple $(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}})$ from $Choices_j$, and all chosen integers k_j (in *Step* 6), with $j \in [n]$, will lead to an output vector that is an image of $\mathbf{in}$ under Ψ.

   Since $\underline{\mathbf{x}}$ and $\underline{\mathbf{y}}$ are corresponding shifts of $\mathbf{id}$ and $\mathbf{in}$, there is an index $t \in [n]$ such that $\sigma_t(\mathbf{id}) = \underline{\mathbf{x}}$ and $\sigma_t(\mathbf{in}) = \underline{\mathbf{y}}$. By the algorithm (*Step* 4), $\underline{\mathbf{z}} \in \Psi(\underline{\mathbf{y}})$. Hence, $\sigma_{-t}(\underline{\mathbf{z}}) \in \sigma_{-t} \circ \Psi(\underline{\mathbf{y}})$. Since Ψ is a circularly symmetric relation, $\sigma_{-t} \circ \Psi \subseteq \Psi \circ \sigma_{-t}$. It follows that $\sigma_{-t}(\underline{\mathbf{z}}) \in \Psi \circ \sigma_{-t}(\underline{\mathbf{y}})$. Since $\sigma_t(\mathbf{in}) = \underline{\mathbf{y}}$, or equivalently $\mathbf{in} = \sigma_{-t}(\underline{\mathbf{y}})$, it follows that $\Psi \circ \sigma_{-t}(\underline{\mathbf{y}}) = \Psi(\sigma_{-t}(\underline{\mathbf{y}})) = \Psi(\mathbf{in})$. Hence, $\sigma_{-t}(\underline{\mathbf{z}}) \in \Psi(\mathbf{in})$. So, it suffices to prove that $\mathbf{out} = \sigma_{-t}(\underline{\mathbf{z}})$.

   By the choice of k_j (*Step* 6/*b*), $\underline{\mathbf{x}} = \sigma_{k_j}(\mathbf{vid}_j) = \sigma_{k_j}(\sigma_j(\mathbf{id})) = \sigma_{k_j+j}(\mathbf{id})$, and $\underline{\mathbf{y}} = \sigma_{k_j}(\mathbf{vin}_j) = \sigma_{k_j}(\sigma_j(\mathbf{in})) = \sigma_{k_j+j}(\mathbf{in})$. Since $\underline{\mathbf{x}} = \sigma_t(\mathbf{id})$ and $\underline{\mathbf{y}} = \sigma_t(\mathbf{in})$, it follows that $\sigma_{k_j+j}(\mathbf{id}) = \sigma_t(\mathbf{id})$ and $\sigma_{k_j+j}(\mathbf{in}) = \sigma_t(\mathbf{in})$. Hence, Lemma 2.3 implies that $(k_j + j) \equiv t \pmod{\mathsf{T}(\mathbf{id})}$ and $(k_j + j) \equiv t \pmod{\mathsf{T}(\mathbf{in})}$. Thus, each of $\mathsf{T}(\mathbf{id})$ and $\mathsf{T}(\mathbf{in})$ divides $k_j + j - t$. Since the Least Common Multiple of two divisors of a number also divides the number, it follows that $\mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))$ divides $k_j + j - t$.

   By *Step* 4, $\mathsf{T}(\underline{\mathbf{z}})$ divides $\mathsf{MB}(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \mathsf{LCM}(\mathsf{T}(\underline{\mathbf{x}}), \mathsf{T}(\underline{\mathbf{y}}))$ (by Lemma 3.1). Since $\underline{\mathbf{x}}$ and $\underline{\mathbf{y}}$ are cyclic shifts of $\mathbf{id}$ and $\mathbf{in}$, respectively, Lemma 2.2 implies that $\mathsf{T}(\underline{\mathbf{x}}) = \mathsf{T}(\mathbf{id})$, and $\mathsf{T}(\underline{\mathbf{y}}) = \mathsf{T}(\mathbf{in})$. So, $\mathsf{T}(\underline{\mathbf{z}})$ divides $\mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))$.

   It follows that $\mathsf{T}(\underline{\mathbf{z}})$ divides $k_j + j - t$, or $(-k_j) \equiv (j - t) \pmod{\mathsf{T}(\underline{\mathbf{z}})}$. Hence, by Lemma 2.3, $\sigma_{-k_j}(\underline{\mathbf{z}}) = \sigma_{j-t}(\underline{\mathbf{z}}) = \sigma_j(\sigma_{-t}(\underline{\mathbf{z}}))$. By *Step* 7, this implies that $\mathbf{out}_j$ is the jth entry of $\sigma_{-t}(\underline{\mathbf{z}})$. Since j was chosen arbitrarily, it follows that $\mathbf{out} = \sigma_{-t}(\underline{\mathbf{z}})$, as needed.

The proof is now complete. ■

We remark that the proof of Proposition 4.2 requires that Ψ be circularly symmetric. It is not evident whether this assumption is *essential*.

4.2. Main results

We first consider the solvability of an arbitrary (circularly symmetric) relation Ψ on an arbitrary set of rings **ID**. We provide a definition of compatibility between a relation Ψ and a set of rings **ID**, as a generalization of Definition 4.1.

Definition 4.2 (*Compatibility with a Set of Rings*). The relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is ***compatible*** with the set of rings $\mathbf{ID} \subseteq \Sigma^n$ if for each input vector $\mathbf{in} \in \mathsf{Dom}(\Psi)$, there is some output vector $\mathbf{out} \in \Psi(\mathbf{in})$ such that $\mathsf{T}(\mathbf{out})$ divides $\mathsf{MB}(\mathbf{ID}, \mathbf{in})$.

Recall that $\mathsf{MB}(\mathbf{ID}, \mathbf{in})$ can be computed efficiently; thus, compatibility with sets of rings can be checked efficiently as well. We prove:

Theorem 4.3 (*Partially Eponymous Solvability Theorem*). *A circularly symmetric relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is solvable on a set of rings $\mathbf{ID} \subseteq \Sigma^n$ if and only if Ψ is compatible with* **ID**.

Proof. Assume first that Ψ is solvable on **ID**. Then, there is a distributed algorithm $\mathcal{A}$ such that for each configuration $\langle \mathbf{id}, \mathbf{in} \rangle \in \mathbf{ID} \times \mathsf{Dom}(\Psi)$, $\mathcal{A}$ solves the set of output vectors $\Psi(\mathbf{in})$ on $\langle \mathbf{id}, \mathbf{in} \rangle$. So, fix the configuration $\langle \widetilde{\mathbf{id}}, \mathbf{in} \rangle$ for an arbitrary vector $\mathbf{in} \in \mathsf{Dom}(\Psi)$. It follows that $\mathcal{A}$ solves the set of output vectors $\Psi(\mathbf{in})$ on $\langle \widetilde{\mathbf{id}}, \mathbf{in} \rangle$; so, $\Psi(\mathbf{in})$ is solvable on $\langle \widetilde{\mathbf{id}}, \mathbf{in} \rangle$. By Proposition 4.1, this implies that $\Psi(\mathbf{in})$ is compatible with $\langle \widetilde{\mathbf{id}}, \mathbf{in} \rangle$. Hence, by definition of compatibility, there is some output vector $\mathbf{out} \in \Psi(\mathbf{in})$ such that $\mathsf{T}(\mathbf{out})$ divides $\mathsf{MB}(\widetilde{\mathbf{id}}, \mathbf{in}) = \mathsf{MB}(\mathbf{ID}, \mathbf{in})$. Since $\mathbf{in} \in \mathsf{Dom}(\Psi)$ was chosen arbitrarily, it follows that Ψ is compatible with **ID**.

Assume now that Ψ is compatible with **ID**. Then, for each input vector $\mathbf{in} \in \mathsf{Dom}(\Psi)$, there is an output vector $\mathbf{out} \in \Psi(\mathbf{in})$ such that $\mathsf{T}(\mathbf{out})$ divides $\mathsf{MB}(\mathbf{ID}, \mathbf{in})$. Fix an arbitrary configuration $\langle \mathbf{id}, \mathbf{in} \rangle \in \mathbf{ID} \times \mathsf{Dom}(\Psi)$. By Lemma 3.2, this implies that there is some output vector $\mathbf{out} \in \Psi(\mathbf{in})$ such that $\mathsf{T}(\mathbf{out})$ divides $\mathsf{MB}(\mathbf{id}, \mathbf{in})$.

Recall the distributed algorithm $\mathcal{A}_\Psi$ from Section 4.1. Clearly, $\langle \mathbf{id}, \mathbf{in} \rangle$ is some entry of the array $Views_j$ for each processor $j \in [n]$. Since $\mathbf{out} \in \Psi(\mathbf{in})$ and $\mathsf{T}(\mathbf{out})$ divides $\mathsf{MB}(\mathbf{id}, \mathbf{in})$, *Step* 4 implies that $(\mathbf{id}, \mathbf{in}, \mathbf{out}) \in Choices_j$ for each processor $j \in [n]$; so, $Choices_j \neq \emptyset$. By *Step* 5, it follows that $\mathcal{A}_\Psi$ does not solve $\{\perp^n\}$ on $\langle \mathbf{id}, \mathbf{in} \rangle$. Hence, Proposition 4.2 implies that $\mathcal{A}_\Psi$ solves $\Psi(\mathbf{in})$ on $\langle \mathbf{id}, \mathbf{in} \rangle$. Since the configuration $\langle \mathbf{id}, \mathbf{in} \rangle \in \mathbf{ID} \times \mathsf{Dom}(\Psi)$ was chosen arbitrarily, it follows that Ψ is solvable on $\mathbf{ID}$. ■

Since compatibility (with sets of rings) is efficiently checkable, Theorem 4.3 provides an efficient characterization of solvability for partially eponymous rings.

We remark that the proof of Theorem 4.3 only uses synchronous executions: the proof only invokes Proposition 4.1, which, in turn, invokes Lemma 3.3, which applies to the restriction to synchronous executions; no asynchronous (but not synchronous) execution is needed for the proof of Theorem 4.3. So, Theorem 4.3 provides an *identical* characterization of solvability for the special case of synchronous, partially eponymous rings. Hence, Theorem 4.3 implies a *collapse* between synchronous and asynchronous partially eponymous rings with respect to solvability of (circularly symmetric) relations on sets of rings. We continue to prove:

Theorem 4.4 (*Partially Eponymous Computability Theorem*). *Algorithm $\mathcal{A}_\Psi$ computes the circularly symmetric relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ on a set of rings $\mathbf{ID} \subseteq \Sigma^n$.*

Proof. Fix an arbitrary configuration $\langle \mathbf{id}, \mathbf{in} \rangle \in \mathbf{ID} \times \mathsf{Dom}(\Psi)$. We will prove that $\mathcal{A}_\Psi$ computes $\Psi(\mathbf{in})$ on $\langle \mathbf{id}, \mathbf{in} \rangle$. We proceed by case analysis.

1. Assume first that $\Psi(\mathbf{in})$ is solvable on $\langle \mathbf{id}, \mathbf{in} \rangle$. By Proposition 4.1, it follows that $\Psi(\mathbf{in})$ is compatible with $\langle \mathbf{id}, \mathbf{in} \rangle$. By definition of compatibility, this implies that there is some output vector $\mathbf{out} \in \Psi(\mathbf{in})$ such that $\mathsf{T}(\mathbf{out})$ divides $\mathsf{MB}(\mathbf{id}, \mathbf{in})$. From *Step* 3, $\langle \mathbf{id}, \mathbf{in} \rangle$ is some entry of the array $Views_j$ for each processor $j \in [n]$. Since $\mathbf{out} \in \Psi(\mathbf{in})$ and $\mathsf{T}(\mathbf{out})$ divides $\mathsf{MB}(\mathbf{id}, \mathbf{in})$, it follows by *Step* 4 that $(\mathbf{id}, \mathbf{in}, \mathbf{out}) \in Choices_j$; so $Choices_j \neq \emptyset$. By *Step* 5, this implies that $\mathcal{A}_\Psi$ does not solve $\{\perp^n\}$ on $\langle \mathbf{id}, \mathbf{in} \rangle$. Hence, Proposition 4.2 implies that $\mathcal{A}_\Psi$ solves $\Psi(\mathbf{in})$ on $\langle \mathbf{id}, \mathbf{in} \rangle$.
2. Assume now that $\Psi(\mathbf{in})$ is not solvable on $\langle \mathbf{id}, \mathbf{in} \rangle$. Hence, in particular, $\mathcal{A}_\Psi$ does not solve $\Psi(\mathbf{in})$ on $\langle \mathbf{id}, \mathbf{in} \rangle$. By Proposition 4.2, it follows that $\mathcal{A}_\Psi$ solves $\{\perp^n\}$ on $\langle \mathbf{id}, \mathbf{in} \rangle$.

By definition of computability, the claim follows. ■

Theorem 4.4 immediately implies:

Corollary 4.5. *Every circularly symmetric relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is computable on a set of rings $\mathbf{ID} \subseteq \Sigma^n$.*

4.3. Applications

We continue with some applications of the Partially Eponymous Solvability Theorem. We prove:

Theorem 4.6 (*Solvability of k-Periodic Relations*). *A total, circularly symmetric, k-periodic relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is solvable on an ℓ-periodic set of rings $\mathbf{ID} \subseteq \Sigma^n$ if and only if k divides ℓ.*

Proof. Since Ψ is k-periodic, we have that for each input vector $\mathbf{in} \in \mathsf{Dom}(\Psi)$, for each output vector $\mathbf{out} \in \Psi(\mathbf{in})$, $\mathsf{T}(\mathbf{out}) = k$. Since $\mathbf{ID}$ is ℓ-periodic, $\mathsf{T}(\widetilde{\mathbf{id}}) = \ell$. For each input vector $\mathbf{in} \in \mathsf{Dom}(\Psi)$,

$$\begin{aligned} \mathsf{MB}(\mathbf{ID}, \mathbf{in}) &= \mathsf{MB}(\widetilde{\mathbf{id}}, \mathbf{in}) && \text{(by definition of } \mathsf{MB}(\mathbf{ID}, \mathbf{in})) \\ &= \mathsf{LCM}(\mathsf{T}(\widetilde{\mathbf{id}}), \mathsf{T}(\mathbf{in})) && \text{(by Lemma 3.1)}. \end{aligned}$$

Assume first that Ψ is solvable on $\mathbf{ID}$. By Theorem 4.3, this implies that Ψ is compatible with $\mathbf{ID}$. Since Ψ is total, fix $\mathbf{in} \in \mathsf{Dom}(\Psi)$ to be constant; so, $\mathsf{T}(\mathbf{in}) = 1$. Hence, $\mathsf{MB}(\mathbf{ID}, \mathbf{in}) = \mathsf{T}(\widetilde{\mathbf{id}}) = \ell$. By the compatibility of Ψ with $\mathbf{ID}$, there is an output vector $\mathbf{out} \in \Psi(\mathbf{in})$ such that $\mathsf{T}(\mathbf{out})$ divides $\mathsf{MB}(\mathbf{ID}, \mathbf{in})$. It follows that k divides ℓ.

Assume now that k divides ℓ. This implies that $\mathsf{T}(\mathbf{out})$ divides $\mathsf{T}(\widetilde{\mathbf{id}})$. Since $\mathsf{MB}(\mathbf{ID}, \mathbf{in}) = \mathsf{LCM}(\mathsf{T}(\widetilde{\mathbf{id}}), \mathsf{T}(\mathbf{in}))$, it follows that $\mathsf{T}(\mathbf{out})$ divides $\mathsf{MB}(\mathbf{ID}, \mathbf{in})$. Hence, Ψ is compatible with $\mathbf{ID}$. Theorem 4.3 implies now that Ψ is solvable on $\mathbf{ID}$. ■

A careful inspection of the proof of Theorem 4.6 reveals that it applies more generally to a (not necessarily total) circularly symmetric, k-periodic relation Ψ such that $\mathsf{Dom}(\Psi)$ includes at least one constant vector. Many natural relations assume no inputs; so, they are viewed either as being total, or as their domain consisting of a single, constant input vector. Theorem 4.6 applies to all such relations. We conclude with two immediate consequences of Theorem 4.6.

Corollary 4.7 (*Solvability of Uniperiodic Relations*). *A total, circularly symmetric, uniperiodic relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is solvable on a set of rings $\mathbf{ID} \subseteq \Sigma^n$.*

Corollary 4.8 (*Solvability of Aperiodic Relations*). *A total, circularly symmetric, aperiodic relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is solvable on a set of rings $\mathbf{ID} \subseteq \Sigma^n$ if and only if $\mathbf{ID}$ is aperiodic.*

$\mathcal{A}_{\mathrm{MLE}}(\alpha)$: CODE FOR PROCESSOR j WITH IDENTITY id_j AND INPUT in_j

Initially, $label_j = segment_j = left\_segment_j = right\_segment_j = \lambda$.

1: Upon receiving message $\langle wake \rangle$ **do**
2: **If** $\alpha = 1$ **then** terminate as a leader.
3: **Else** send message $\langle probe, 0, 1 \rangle$ to left.

4: Upon receiving message $\langle probe, r, d \rangle$ from right **do**
5: **If** $d < 2^r$ **then** send message $\langle probe, r, d+1 \rangle$ to left.
6: **If** $d = 2^r$ **then** send message $\langle reply, \langle (id_j, in_j) \rangle, r, 1 \rangle$ to right.

7: Upon receiving message $\langle reply, received\_seg, r, d \rangle$ from left **do**
8: **If** $d < 2^r$ **then** send message $\langle reply, received\_seg \diamond \langle (id_j, in_j) \rangle, r, d+1 \rangle$ to right.
9: **If** $d = 2^r$ **then do**
10: $left\_segment_j := received\_seg$.
11: Send message $\langle probe, r, 1 \rangle$ to right.

12: Upon receiving message $\langle probe, r, d \rangle$ from left **do**
13: **If** $d < 2^{r+1} - 1$ **then** send message $\langle probe, r, d+1 \rangle$ to right.
14: **If** $d = 2^{r+1} - 1$ **then** send message $\langle reply, \langle (id_j, in_j) \rangle, r, 1 \rangle$ to left.

15: Upon receiving message $\langle reply, received\_seg, r, d \rangle$ from right **do**
16: **If** $d < 2^{r+1} - 1$ **then** send message $\langle reply, \langle (id_j, in_j) \rangle \diamond received\_seg, r, d+1 \rangle$ to left.
17: **If** $d = 2^{r+1} - 1$ **then do**
18: $right\_segment_j := received\_seg$.
19: $segment_j := left\_segment_j \diamond \langle (id_j, in_j) \rangle \diamond right\_segment_j$.
20: $label_j := \pi_{2^r}(\sigma_{2^r}(segment_j))$.
21: **If** for each $i \in [2^r]$, $label_j \prec \pi_{2^r}(\sigma_i(segment_j))$ **and** $label_j \preceq \pi_{2^r}(\sigma_{2^r+i+1}(segment_j))$, **then do**
22: **If** $2^{r+1} \geq \alpha$ **then** terminate as a leader.
23: **Else** send message $\langle probe, r+1, 1 \rangle$ to left.
24: **Else** terminate as a non-leader.

Fig. 2. Algorithm $\mathcal{A}_{\mathrm{MLE}}(\alpha)$: code for processor j with advice α.

5. Message complexity

5.1. Multiple Leader Election

We present an asynchronous distributed algorithm $\mathcal{A}_{\mathrm{MLE}}(\alpha)$ with advice α. Here, α is an integer that is available to each processor, and it will act as a parameter. $\mathcal{A}_{\mathrm{MLE}}(\alpha)$ will satisfy a particular correctness property for certain specific values of advice α; furthermore, the message complexity of $\mathcal{A}_{\mathrm{MLE}}(\alpha)$ will depend on α. The distributed algorithm $\mathcal{A}_{\mathrm{MLE}}(\alpha)$ appears in pseudocode in Fig. 2. We start with an informal description of the algorithm $\mathcal{A}_{\mathrm{MLE}}(\alpha)$.

Each processor explores neighborhoods around it whose sizes double in each ***phase***. A processor ***enters*** a phase when it initiates the first message tagged with the phase number; it enters phase 0 at *Step* 3, and phase $r+1$, with $r \geq 0$, at *Step* 23. Before entering any phase, including phase 0, a processor checks to see if it should terminate as a leader (*Steps* 2 and 22). Thus, a processor may terminate before even entering phase 0 without sending any messages, exactly when $\alpha = 1$.

When it enters phase r, a processor collects the identities and inputs of other processors in its neighborhood that are 2^r to the left or $2^{r+1} - 1$ to the right. It then uses this information to locally construct the 2^{r+1} prefixes of length 2^r of the views of all processors that are 2^r to the left or to the right of it. The processor proceeds to compare the prefix of length 2^r of its own view to those prefixes it has constructed (*Step* 21); it ***survives*** the phase (and does not terminate as a non-leader) if its own prefix is *the lexicographically least* among all 2^{r+1} constructed prefixes of the views of its neighbors. Finally, it proceeds iteratively to check whether it should enter the next phase.

Note that by *Steps* 2 and 22, a processor enters phase r if and only if $2^r < \alpha$; thus, α determines the size of the largest neighborhood that each processor may explore before terminating. Note also that a processor enters a phase if and only if during the previous (if any) phase, the processor terminated neither as a leader nor as a non-leader. Thus, a processor terminates as a leader if and only if it survives the last phase (with $r = \lceil \lg \alpha \rceil - 1$).

We proceed to prove the message complexity and correctness properties of $\mathcal{A}_{\mathrm{MLE}}(\alpha)$. We start with a simple fact:

Lemma 5.1. *Consider a processor $j \in [n]$ that entered phase r of $\mathcal{A}_{\mathrm{MLE}}(\alpha)$. Then, when processor j is in* Step 21,

$$segment_j = \pi_{3 \cdot 2^r}(\sigma_{-2^r}(view_j(\mathbf{id}, \mathbf{in})))$$

and

$$label_j = \pi_{2^r}(view_j(\mathbf{id}, \mathbf{in})).$$

Proof. Upon entering phase r, processor j initiated a *probe* message to its left (*Step* 3 if $r = 0$, and *Step* 23 otherwise). This *probe* message was received (*Step* 4) by processors to the left of processor j, and it was forwarded (*Step* 5) until the message reached the 2^rth processor $j - 2^r$ to the left of processor j. At that point, processor $j - 2^r$ initiated a *reply* message to its right (*Step* 6); the *reply* message travelled towards processor j collecting the identities and inputs of the processors it encountered.

This *reply* message was received (*Step* 7) by processors to the left of processor j, and it was forwarded (*Step* 8) until the message reached the 2^rth processor $j - 2^r + 2^r = j$ to the right of processor $j - 2^r$ (*Step* 9). At that point, processor j set $left\_segment_j$ to the vector of received identities and inputs (*Step* 10). Clearly, by construction, $left\_segment_j$ is a prefix of length 2^r of the view of processor $j - 2^r$. So,

$$left\_segment_j = \pi_{2^r}(view_{j-2^r}(\mathbf{id}, \mathbf{in})).$$

Processor j then initiated a *probe* message to its right (*Step* 11). This *probe* message was received (*Step* 12) by processors to the right of processor j, and it was forwarded (*Step* 13) until the message reached the $(2^{r+1} - 1)$th processor $j + (2^{r+1} - 1)$ to the right of processor j. At that point, processor $j + (2^{r+1} - 1)$ initiated a *reply* message to its left (*Step* 14); the *reply* message travelled towards processor j collecting the identities and inputs of the processors it encountered.

This *reply* message was received (*Step* 15) by processors to the right of processor j, and it was forwarded (*Step* 16) until the message reached the $(2^{r+1} - 1)$th processor $j + (2^{r+1} - 1) - (2^{r+1} - 1) = j$ to the left of processor $j + (2^{r+1} - 1)$ (*Step* 17). At that point, processor j set $right\_segment_j$ to the vector of received identities and inputs (*Step* 18). Clearly, by construction, $right\_segment_j$ is a prefix of length $(2^{r+1} - 1)$ of the view of processor $j + 1$. So,

$$right\_segment_j = \pi_{(2^{r+1}-1)}(view_{j+1}(\mathbf{id}, \mathbf{in})).$$

Processor j then set $segment_j$ to $left\_segment_j \diamond \langle (id_j, in_j) \rangle \diamond right\_segment_j$ (*Step* 19). Overall, $segment_j$ is a prefix of length $3 \cdot 2^r$ of the view of processor $j - 2^r$. Thus,

$$\begin{aligned} segment_j &= \pi_{3 \cdot 2^r}(view_{j-2^r}(\mathbf{id}, \mathbf{in})) \\ &= \pi_{3 \cdot 2^r}(\sigma_{-2^r}(view_j(\mathbf{id}, \mathbf{in}))) && \text{(by definition of view).} \end{aligned}$$

Finally, observe that (i) $left\_segment_j$ has length 2^r, and (ii) $\langle (id_j, in_j) \rangle \diamond right\_segment_j$ has length 2^{r+1}. Recall also that $right\_segment_j$ is a prefix of $view_{j+1}(\mathbf{id}, \mathbf{in})$. This immediately implies that (iii) $\langle (id_j, in_j) \rangle \diamond right\_segment_j$ is a prefix of $view_j(\mathbf{id}, \mathbf{in})$. Then,

$$\begin{aligned} label_j &= \pi_{2^r}(\sigma_{2^r}(segment_j)) && \text{(by } \textit{Step} \text{ 20)} \\ &= \pi_{2^r}(\sigma_{2^r}(left\_segment_j \diamond \langle (id_j, in_j) \rangle \diamond right\_segment_j)) && \text{(by } \textit{Step} \text{ 19)} \\ &= \pi_{2^r}(\langle (id_j, in_j) \rangle \diamond right\_segment_j \diamond left\_segment_j) && \text{(by observation (i))} \\ &= \pi_{2^r}(\langle (id_j, in_j) \rangle \diamond right\_segment_j) && \text{(by observation (ii))} \\ &= \pi_{2^r}(view_j(\mathbf{id}, \mathbf{in})) && \text{(by observation (iii))}, \end{aligned}$$

as needed. ■

We continue to prove:

Lemma 5.2. *Consider a processor $j \in [n]$ that entered phase r of $\mathcal{A}_{\mathrm{MLE}}(\alpha)$. Then, processor j survives phase r if and only if for each $i \in [2^r]$, $label_j \prec label_{j-(2^r-i)}$ and $label_j \preceq label_{j+(i+1)}$.*

Proof. Note that processor j survives phase r if and only if the predicate checked in *Step* 21 during phase r is true. The predicate is true if and only if for every $i \in [2^r]$,

$$
\begin{aligned}
label_j &\prec \pi_{2^r}(\sigma_i(segment_j)) && \text{(by } \textit{Step } 21\text{)}\\
&= \pi_{2^r}(\sigma_i(\pi_{3.2^r}(\sigma_{-2^r}(view_j(\mathbf{id},\mathbf{in}))))) && \text{(by Lemma 5.1)}\\
&= \pi_{2^r}(\sigma_{-(2^r-i)}(view_j(\mathbf{id},\mathbf{in}))) && \text{(by Lemma 2.1; } 2^r \le 3\cdot 2^r - i\text{)}\\
&= \pi_{2^r}(view_{j-(2^r-i)}(\mathbf{id},\mathbf{in})) && \text{(by definition of view)}\\
&= label_{j-(2^r-i)} && \text{(by Lemma 5.1)}
\end{aligned}
$$

and

$$
\begin{aligned}
label_j &\preceq \pi_{2^r}(\sigma_{2^r+i+1}(segment_j)) && \text{(by } \textit{Step } 21\text{)}\\
&= \pi_{2^r}(\sigma_{2^r+i+1}(\pi_{3.2^r}(\sigma_{-2^r}(view_j(\mathbf{id},\mathbf{in}))))) && \text{(by Lemma 5.1)}\\
&= \pi_{2^r}(\sigma_{i+1}(view_j(\mathbf{id},\mathbf{in}))) && \text{(by Lemma 2.1; } 2^r \le 3\cdot 2^r - (2^r+i+1)\text{)}\\
&= \pi_{2^r}(view_{j+(i+1)}(\mathbf{id},\mathbf{in})) && \text{(by definition of view)}\\
&= label_{j+(i+1)} && \text{(by Lemma 5.1)}\,,
\end{aligned}
$$

and the claim follows. ■

We now prove an upper bound on the number of processors surviving phase r.

Lemma 5.3. *At most $\frac{n}{2^r+1}$ processors survive phase r of $\mathcal{A}_{\mathrm{MLE}}(\alpha)$.*

Proof. It suffices to prove that any given processor $j \in [n]$ survives phase r only if no processor in the 2^r-neighborhood of processor j survives phase r. Assume, by way of contradiction, that there is a processor $j' \neq j$ in the 2^r-neighborhood of processor j such that both processors j' and j survive phase r. Since $j' \in \{j-2^r, \ldots, j-1, j+1, \ldots, j+2^r\}$, there is an index $i \in [2^r]$ such that either $j' = j-(2^r-i)$ or $j' = j+(i+1)$.

- Assume first that $j' = j-(2^r-i)$. Lemma 5.2 implies that $label_j \prec label_{j'}$. Note that $j = j'+(2^r-i) = j'+(i'+1)$ for $i' = 2^r-i-1 \in [2^r]$. Hence, Lemma 5.2 implies that $label_{j'} \preceq label_j$. A contradiction.
- Assume now that $j' = j+(i+1)$. Lemma 5.2 implies that $label_j \preceq label_{j'}$. Note that $j = j'-(i+1) = j'-(2^r-i')$ for $i' = 2^r-i-1 \in [2^r]$. Hence, Lemma 5.2 implies that $label_{j'} \prec label_j$. A contradiction.

Since we obtained a contradiction in all possible cases, the claim follows. ■

We now analyze the number of messages sent by algorithm $\mathcal{A}_{\mathrm{MLE}}(\alpha)$.

Proposition 5.4. *Algorithm $\mathcal{A}_{\mathrm{MLE}}(\alpha)$ sends $\mathcal{O}(n\cdot\lg\alpha)$ messages on a ring of size n.*

Proof. By *Steps* 2 and 22, the algorithm executes for phases 0 to $\lceil\lg\alpha\rceil - 1$. By Lemma 5.3, at most $\frac{n}{2^r+1}$ processors survive phase r. At phase r, where $1 \le r \le \lceil\lg\alpha\rceil - 1$, each non-terminated processor (out of those $\frac{n}{2^{r-1}+1}$ that survived phase $r-1$) initiates less than $6\cdot 2^r$ messages ($2\cdot 2^r$ to its left and $2\cdot(2^{r+1}-1)$ to its right); at phase 0, each processor initiates at most 4 messages. So, the total number of messages is less than

$$
\begin{aligned}
4n + \sum_{r=1}^{\lceil\lg\alpha\rceil-1}\left(\frac{n}{2^{r-1}+1}\right)\cdot 6\cdot 2^r &= 4n + 12n\sum_{r=1}^{\lceil\lg\alpha\rceil-1}\left(1-\frac{1}{2^{r-1}+1}\right)\\
&< 4n + 12n\,(\lceil\lg\alpha\rceil - 1)\\
&= \mathcal{O}(n\cdot\lg\alpha)\,,
\end{aligned}
$$

as needed. ■

We continue with a correctness property of the algorithm $\mathcal{A}_{\mathrm{MLE}}(\alpha)$ for *specific* values of advice α. For any $\alpha \in \mathbb{Z}^+$, consider the function $\Phi_\alpha : \Sigma^n \to \mathbb{Z}^+$ such that $\Phi_\alpha(\mathbf{in}) = \lceil\frac{2n}{\alpha}\rceil$ for each input vector $\mathbf{in} \in \Sigma^n$. We prove:

Proposition 5.5. *Algorithm $\mathcal{A}_{\mathrm{MLE}}(\alpha)$ with advice α, where $1 \le \alpha \le \mathsf{MB}(\mathbf{id},\mathbf{in})$, solves the set of output vectors Φ_α-LE$(\mathbf{in})$ on the configuration $\langle\mathbf{id},\mathbf{in}\rangle \in \Sigma^n \times \Sigma^n$.*

Proof. We start with an informal outline of the proof. We first prove in Part A that at least one processor will terminate as a leader; this will be the processor with the least view under lexicographic ordering. We will establish that this processor will survive every phase that it enters and be elected as leader. We then prove in Part B that at most $\Phi_\alpha(\mathbf{in}) = \lceil\frac{2n}{\alpha}\rceil$ processors will terminate as leaders in the last phase. We now continue with the details of the formal proof.

Part A: Fix a processor j with the least view under lexicographic ordering. Since each shift of $view_j(\mathbf{id},\mathbf{in})$ is the view of some processor, it follows that $view_j(\mathbf{id},\mathbf{in})$ is shift-minimal.

Fix an arbitrary phase $r \ge 0$ such that processor j enters phase r. If $r = 0$, then *Step* 2 implies that $\alpha \ge 2$. If $r > 0$, then *Step* 22 implies that $2^r < \alpha$. In either case, it follows that $2^r < \alpha$. Then,

$$
\begin{aligned}
2^r &< \alpha\\
&\le \mathsf{MB}(\mathbf{id},\mathbf{in}) && \text{(by assumption)}\\
&= \mathsf{LCM}(\mathsf{T}(\mathbf{id}),\mathsf{T}(\mathbf{in})) && \text{(by Lemma 3.1)}\,.
\end{aligned}
$$

Fix an index $i \in [2^r]$; we will examine the processors $j - (2^r - i)$ and $j + (i + 1)$ in the left and right segment of the 2^r-neighborhood of processor j, respectively.

We will use Lemma 2.6 with $view_j(\mathbf{id}, \mathbf{in})$ for $\mathbf{x}$. Since $view_j(\mathbf{id}, \mathbf{in})$ is shift-minimal, the assumption in Lemma 2.6 holds. Recall that $\mathsf{T}(view_j(\mathbf{id}, \mathbf{in})) = \mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))$ (by Lemmas 2.2 and 2.7).

- Use first 2^r for ℓ and $i + 1$ for m. Then,

$$\begin{aligned} label_j &= \pi_{2^r}(view_j(\mathbf{id}, \mathbf{in})) && \text{(by Lemma 5.1)} \\ &\preceq \pi_{2^r}(\sigma_{i+1}(view_j(\mathbf{id}, \mathbf{in}))) && \text{(by Lemma 2.6 (Condition (1)))} \\ &= \pi_{2^r}(view_{j+(i+1)}(\mathbf{id}, \mathbf{in})) && \text{(by definition of view)} \\ &= label_{j+(i+1)} && \text{(by Lemma 5.1)}\ . \end{aligned}$$

- Use now 2^r for ℓ and $2^r - i$ for m, so that $\ell \geq m$. Since $i \in [2^r]$, and $2^r < \mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))$, it follows that $1 \leq 2^r - i < \mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))$. Hence, $(2^r - i) \not\equiv 0 \pmod{\mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))}$, so that the assumption for Condition (2) holds. It follows that

$$\begin{aligned} label_j &= \pi_{2^r}(view_j(\mathbf{id}, \mathbf{in})) && \text{(by Lemma 5.1)} \\ &\prec \pi_{2^r}(\sigma_{-(2^r-i)}(view_j(\mathbf{id}, \mathbf{in}))) && \text{(by Lemma 2.6 (Condition (2)))} \\ &= \pi_{2^r}(view_{j-(2^r-i)}(\mathbf{id}, \mathbf{in})) && \text{(by definition of view)} \\ &= label_{j-(2^r-i)} && \text{(by Lemma 5.1)}\ . \end{aligned}$$

Since $label_j \preceq label_{j+(i+1)}$ and $label_j \prec label_{j-(2^r-i)}$, and $i \in [2^r]$ was chosen arbitrarily, Lemma 5.2 implies that processor j survives phase r. Since r was chosen arbitrarily, it follows that processor j survives every phase that it enters; so, *Step* 24 is *never* executed. Thus, processor j terminates as a leader (either in *Step* 2 or in *Step* 22).

Part B: Consider the last phase r with $r = \lceil \lg \alpha \rceil - 1$ (*Step* 22). By Lemma 5.3, at most $\frac{n}{2^r+1} \leq \frac{n}{\frac{\alpha}{2}+1} < \left\lceil \frac{2n}{\alpha} \right\rceil = \Phi_\alpha(\mathbf{in})$ processors terminate as leaders, and the proof is now complete. ■

5.2. Universal upper bound

We now prove:

Theorem 5.6 (*Partially Eponymous Message Complexity Theorem*). *The circularly symmetric relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is computable on a set of rings $\mathbf{ID} \subseteq \Sigma^n$ with*

$$\mathcal{O}\left(\frac{n^2}{\mathsf{LMB}(\mathbf{ID}, \Psi)} + n \cdot \lg \mathsf{LMB}(\mathbf{ID}, \Psi)\right)$$

messages.

Proof. Fig. 3 depicts a distributed algorithm $\mathcal{A}_\Psi{}'$ (which is an instantiation of $\mathcal{A}_\Psi$) to compute Ψ on $\mathbf{ID}$ with the required number of messages. Fix a configuration $\langle \mathbf{id}, \mathbf{in} \rangle \in \mathbf{ID} \times \Sigma^n$. Since algorithm $\mathcal{A}_\Psi{}'$ is an instantiation of $\mathcal{A}_\Psi$, it suffices to establish that processors construct their views in *Step* 2 with the required number of messages.

In *Step* $2/a$ the distributed algorithm $\mathcal{A}_{\mathsf{MLE}}(\alpha)$ is executed with advice $\alpha = \mathsf{LMB}(\mathbf{ID}, \Psi)$. Since $\alpha = \mathsf{LMB}(\mathbf{ID}, \Psi) \leq \mathsf{MB}(\mathbf{id}, \mathbf{in})$, Proposition 5.5 implies that at least 1 and at most $\left\lceil \frac{2n}{\mathsf{LMB}(\mathbf{ID}, \Psi)} \right\rceil$ processors terminate as leaders. All leaders execute the algorithm $\mathcal{A}_{\mathsf{CV}}$ (see Section 4) in *Step* $2/c$ to construct their views; the rest of the processors forward messages appropriately during this step.

Lemma 5.7. *In* Step 2*, every processor constructs its view, and sends exactly one info message.*

Proof. Assume, by way of contradiction, that there is some processor that does not satisfy the claim. Among all such processors, consider a processor j with the least distance to a leader in the left segment of j's neighborhood; at least one leader was elected, so this processor is well defined.

Observe that processor j is not a leader, since by *Step* $2/c$, all leaders construct their views, and send exactly one *info* message in *Step* $2/d$. So, $j - 1$ is either a leader, or a processor with less distance to a leader in the left segment of $j - 1$'s neighborhood; hence, processor $j - 1$ satisfies the claim. It follows that processor $j - 1$ constructs its view during *Step* 2 and sends exactly one *info* message. Then, processor j receives an *info* message with processor $j - 1$'s view exactly once (*Step* $2/e$). In turn, processor j proceeds to construct its own view by shifting the received view *received_view* one place anti-clockwise (*Step* $2/f$); it sends exactly one *info* message with its own view in *Step* $2/g$. A contradiction to the assumption that processor j did not satisfy the claim. ■

$\mathcal{A}_\Psi'$: CODE FOR PROCESSOR j WITH IDENTITY id_j AND INPUT in_j

1: Upon receiving message $\langle wake \rangle$ **do**

2:
- $\backslash*$ Construct $view_j(\mathbf{id}, \mathbf{in}) = \mathbf{vid}_j \,\|\, \mathbf{vin}_j$. $*\backslash$ $\quad \backslash* \mathbf{vid}_j = \sigma_j(\mathbf{id}), \mathbf{vin}_j = \sigma_j(\mathbf{in}) *\backslash$
- (a) Execute the distributed algorithm $\mathcal{A}_{\mathrm{MLE}}(\alpha)$ with advice $\alpha = \mathsf{LMB}(\mathbf{ID}, \Psi)$.
- (b) **If** elected as leader **then do**
- (c) Execute the distributed algorithm $\mathcal{A}_{\mathrm{CV}}$ to construct $view_j(\mathbf{id}, \mathbf{in})$.
- (d) Send message $\langle info, view_j(\mathbf{id}, \mathbf{in}) \rangle$ to right.
- (e) **Else** upon receiving message $\langle info, received\_view \rangle$ from left **do**
- (f) $view_j(\mathbf{id}, \mathbf{in}) := \sigma_1(received\_view)$.
- (g) Send message $\langle info, view_j(\mathbf{id}, \mathbf{in}) \rangle$ to right.

3: $Views_j[i] := \langle \sigma_i(\mathbf{vid}_j), \sigma_i(\mathbf{vin}_j) \rangle$ for each $i \in [n]$, where n is the size of $view_j(\mathbf{id}, \mathbf{in})$.

4: $Choices_j := \{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \mid \langle \mathbf{x}, \mathbf{y} \rangle \in Views_j;\ \mathbf{z} \in \Psi(\mathbf{y});\ \mathsf{T}(\mathbf{z})$ divides $\mathsf{MB}(\mathbf{x}, \mathbf{y})\}$.

5: **If** $Choices_j = \emptyset$ **then** $out_j := \perp$ and terminate.

6: (a) $(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}}) := \min\{Choices_j\}$;
(b) $k_j := \min\left\{i \in [n] \mid Views_j[i] = \langle \underline{\mathbf{x}}, \underline{\mathbf{y}} \rangle\right\}$, where n is the size of $view_j(\mathbf{id}, \mathbf{in})$.

7: Set out_j to the first entry of $\sigma_{-k_j}(\underline{\mathbf{z}})$ and terminate.

Fig. 3. Algorithm $\mathcal{A}_\Psi'$: code for processor j.

It remains to show that the number of messages is as claimed. By Proposition 5.4, *Step* $2/a$ contributes $\mathcal{O}(n \cdot \lg \mathsf{LMB}(\mathbf{ID}, \Psi))$ messages. *Step* $2/c$ is only executed by the elected leaders, which are at most $\left\lceil \frac{2n}{\mathsf{LMB}(\mathbf{ID}, \Psi)} \right\rceil$; each leader contributes n messages when constructing its view, for a total of $n \cdot \left\lceil \frac{2n}{\mathsf{LMB}(\mathbf{ID}, \Psi)} \right\rceil$ messages. Clearly, there are n *info* messages sent in *Steps* $2/d$ and $2/g$, since each processor sends exactly one such message (by Lemma 5.7). Thus, the total number of messages sent in *Step* 2 is

$$\mathcal{O}(n \cdot \lg \mathsf{LMB}(\mathbf{ID}, \Psi)) + n \cdot \left\lceil \frac{2n}{\mathsf{LMB}(\mathbf{ID}, \Psi)} \right\rceil + n = \mathcal{O}\left(\frac{n^2}{\mathsf{LMB}(\mathbf{ID}, \Psi)} + n \cdot \lg \mathsf{LMB}(\mathbf{ID}, \Psi) \right),$$

as needed. ■

Notice that $\mathcal{A}_\Psi'$ invokes the non-uniform distributed algorithm $\mathcal{A}_{\mathrm{CV}}$ for constructing views. Hence, $\mathcal{A}_\Psi'$ is non-uniform as well. Finally, an inspection of all algorithms used inside $\mathcal{A}_\Psi'$ (namely, $\mathcal{A}_{\mathrm{MLE}}$ and $\mathcal{A}_{\mathrm{CV}}$) reveals that every message sent by $\mathcal{A}_\Psi'$ includes $\mathcal{O}(n)$ identities and inputs. Hence, the number of identities and inputs communicated by $\mathcal{A}_\Psi'$ is $\mathcal{O}(n)$ times its message complexity. Recall that the bit complexity of a single identity or input may not be bounded by some function of n; hence, the bit complexity of $\mathcal{A}_\Psi'$ may not be so bounded as well.

5.3. Applications

We now identify two special classes of sets of rings $\mathbf{ID} \subseteq \Sigma^n$ where the upper bound on message complexity from Theorem 5.6 drops to $\mathcal{O}(n \cdot \lg n)$.

5.3.1. Universal sets of rings

A set of rings $\mathbf{ID} \subseteq \Sigma^n$ is ***universal*** if *LE* is solvable on **ID**. Clearly, every (circularly symmetric) relation is solvable on **ID**. We prove:

Theorem 5.8 (*Message Complexity on Universal Set of Rings*). *A circularly symmetric relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is computable on a universal set of rings $\mathbf{ID} \subseteq \Sigma^n$ with $\mathcal{O}(n \cdot \lg n)$ messages.*

Proof. Since *LE* is solvable on **ID**, and *LE* is circularly symmetric, aperiodic and total, Corollary 4.8 implies that **ID** is aperiodic. So, for each $\mathbf{id} \in \mathbf{ID}$, $\mathsf{T}(\mathbf{id}) = n$. By Lemma 2.4 (Condition (1)), for each $\mathbf{in} \in \mathsf{Dom}(\Psi) \subseteq \Sigma^n$, $\mathsf{T}(\mathbf{in})$ divides n. Hence, for each configuration $\langle \mathbf{id}, \mathbf{in} \rangle \in \mathbf{ID} \times \mathsf{Dom}(\Psi)$,

$$\begin{aligned} \mathsf{MB}(\mathbf{id}, \mathbf{in}) &= \mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in})) && \text{(by Lemma 3.1)} \\ &= n. \end{aligned}$$

Hence, $\mathsf{LMB}(\mathbf{ID}, \Psi) = \min\{\mathsf{MB}(\mathbf{id}, \mathbf{in}) \mid \langle \mathbf{id}, \mathbf{in} \rangle \in \mathbf{ID} \times \mathsf{Dom}(\Psi)\} = n$. Thus, by Theorem 5.6, Ψ is computable on **ID** with $\mathcal{O}\left(\frac{n^2}{n} + n \cdot \lg n\right) = \mathcal{O}(n \cdot \lg n)$ messages, as needed. ■

5.3.2. Multiplicity-bounded sets of rings

A set of rings $\mathbf{ID} \subseteq \Sigma^n$ is μ-***bounded*** if $\max\{\mathsf{M}(\mathbf{ID})\} = \mu$; so, all identity multiplicities are bounded by μ. Clearly, in a μ-bounded set of rings **ID**, $\mathsf{GCD}(\mathsf{M}(\mathbf{ID})) \leq \mu$. We prove a preliminary property of μ-bounded sets of rings.

Lemma 5.9. *Consider a circularly symmetric relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ and a μ-bounded set of rings $\mathbf{ID} \subseteq \Sigma^n$. Then,* $\mathsf{LMB}(\mathbf{ID}, \Psi) \geq \frac{n}{\mu}$.

Proof. Fix a configuration $\langle \mathbf{id}, \mathbf{in} \rangle \in \mathbf{ID} \times \mathsf{Dom}(\Psi)$. By Lemma 2.5 (Condition (2)), $\mathsf{T}(\widetilde{\mathbf{id}})$ divides $\mathsf{T}(\mathbf{id})$; so, $\mathsf{T}(\widetilde{\mathbf{id}})$ divides $\mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))$. Since $\mathsf{MB}(\mathbf{id}, \mathbf{in}) = \mathsf{LCM}(\mathsf{T}(\mathbf{id}), \mathsf{T}(\mathbf{in}))$ (by Lemma 3.1), it follows that $\mathsf{T}(\widetilde{\mathbf{id}})$ divides $\mathsf{MB}(\mathbf{id}, \mathbf{in})$; in particular, $\mathsf{MB}(\mathbf{id}, \mathbf{in}) \geq \mathsf{T}(\widetilde{\mathbf{id}})$. By Lemma 2.5 (Condition (1)), $\mathsf{T}(\widetilde{\mathbf{id}}) = \frac{n}{\mathsf{GCD}(\mathsf{M}(\mathbf{ID}))}$. Since $\mathsf{GCD}(\mathsf{M}(\mathbf{ID})) \leq \mu$, it follows that $\mathsf{T}(\widetilde{\mathbf{id}}) \geq \frac{n}{\mu}$. Hence, $\mathsf{MB}(\mathbf{id}, \mathbf{in}) \geq \frac{n}{\mu}$. Definition 3.5 now implies that $\mathsf{LMB}(\mathbf{ID}, \Psi) \geq \frac{n}{\mu}$. ■

We finally prove:

Theorem 5.10 (*Message Complexity on μ-Bounded Set of Rings*). *A circularly symmetric relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is computable on a μ-bounded set of rings $\mathbf{ID} \subseteq \Sigma^n$ with $\mathcal{O}(n \cdot \max\{\mu, \lg n\})$ messages.*

Proof. By Lemma 5.9, $\mathsf{LMB}(\mathbf{ID}, \Psi) \geq \frac{n}{\mu}$. Recall also that $\mathsf{LMB}(\mathbf{ID}, \Psi) \leq n$, Thus, by Theorem 5.6, Ψ is computable on **ID** with $\mathcal{O}\left(\frac{n^2}{\frac{n}{\mu}} + n \cdot \lg n\right) = \mathcal{O}(n \cdot \mu + n \cdot \lg n) = \mathcal{O}(n \cdot \max\{\mu, \lg n\})$ messages, as needed. ■

We conclude with a special case of μ-bounded sets of rings. Say that a set of rings $\mathbf{ID} \subseteq \Sigma^n$ is ***logarithmic*** if it is μ-bounded with $\mu = \mathcal{O}(\lg n)$. Theorem 5.10 immediately implies:

Corollary 5.11 (*Message Complexity on Logarithmic Set of Rings*). *A circularly symmetric relation $\Psi \subseteq \Sigma^n \times \Sigma^n$ is computable on a logarithmic set of rings $\mathbf{ID} \subseteq \Sigma^n$ with $\mathcal{O}(n \cdot \lg n)$ messages.*

6. Open problems

We presented a comprehensive study of solvability, computability and message complexity for the partially eponymous ring. Our work poses far more interesting questions than it answers. For example, is there a *matching* lower bound to the universal upper bound on message complexity from Theorem 5.6? Can we characterize the class of sets of rings of size n for which this (universal) upper bound becomes $\mathcal{O}(n \cdot \lg n)$? Finally, a challenging task is to extend the results obtained for the partially eponymous ring to other network architectures (such as *cliques*, *hypercubes*, and *tori*).

Acknowledgements

We thank the anonymous *OPODIS 2006* and *Theoretical Computer Science* reviewers for helpful comments.

References

[1] D. Angluin, Local and global properties in networks of processors, in: Proceedings of the 12th Annual ACM Symposium on Theory of Computing, May 1980, pp. 82–93.
[2] H. Attiya, M. Snir, M. Warmuth, Computing on an anonymous ring, Journal of the ACM 35 (4) (1988) 845–875.
[3] P. Boldi, S. Vigna, An effective characterization of computability in anonymous networks, in: J. Welch (Ed.), Proceedings of the 15th International Symposium on Distributed Computing, in: Lecture Notes in Computer Science, vol. 2180, Springer-Verlag, 2001, pp. 33–47. October.
[4] P. Boldi, S. Vigna, Fibrations of graphs, Discrete Mathematics 243 (1–3) (2002) 21–66.
[5] J.E. Burns, A formal model for message passing systems, Technical Report TR-91, Department of Computer Science, Indiana University, September 1980.
[6] J. Chalopin, S. Das, N. Santoro, Groupings and pairings in anonymous networks, in: S. Dolev (Ed.), Proceedings of the 20th International Symposium on Distributed Computing, in: Lecture Notes in Computer Science, vol. 4167, Springer-Verlag, 2006, pp. 105–119. September.
[7] S. Dobrev, A. Pelc, Leader election in rings with nonunique labels, Fundamenta Informaticae 59 (4) (2004) 333–347.
[8] P. Flocchini, E. Kranakis, D. Krizanc, F.L. Luccio, N. Santoro, Sorting and election in anonymous asynchronous rings, Journal of Parallel and Distributed Computing 64 (2) (2004) 254–265.
[9] D. Hirschberg, J.B. Sinclair, Decentralized extrema-finding in circular configurations of processes, Communications of the ACM 23 (11) (1980) 627–628.
[10] F.T. Leighton, Finite common coverings of graphs, Journal of Combinatorial Theory (Series B) 33 (3) (1982) 231–238.
[11] G. LeLann, Distributed systems — towards a formal approach, in: B. Gilchrist (Ed.), Information Processing 77 — Proceedings of the IFIP Congress, August 1977, pp. 155–160.
[12] M. Yamashita, T. Kameda, Computing on anonymous networks, Part I: Characterizing the solvable cases, IEEE Transactions on Parallel and Distributed Systems 7 (1) (1996) 69–89.
[13] M. Yamashita, T. Kameda, Computing on anonymous networks, Part II: Decision and membership problems, IEEE Transactions on Parallel and Distributed Systems 7 (1) (1996) 90–96.
[14] M. Yamashita, T. Kameda, Computing functions on asynchronous anonymous networks, Mathematical Systems Theory 29 (4) (1998) 331–356. Erratum in Theory of Computing Systems 31 (1) (1998) 109.
[15] M. Yamashita, T. Kameda, Leader election problem on networks in which processor identity numbers are not distinct, IEEE Transactions on Parallel and Distributed Systems 10 (9) (1999) 878–887.